

Dynamic Task Sharing for Flexible Human-Robot Teaming under Partial Workspace Observability

Flexible Mensch-Roboter-Zusammenarbeit durch dynamische
Aufgabenteilung unter partieller Arbeitsraumbeobachtbarkeit

Von der Universität Bayreuth
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

von
Dominik Riedelbauch
aus Marktredwitz

1. Gutachter: Prof. Dr. Dominik Henrich
2. Gutachter: Prof. Dr.-Ing. Kirsten Tracht
3. Gutachter: Prof. Dr.-Ing. Bernd Kuhlenkötter

Tag der Einreichung: 12. Juni 2020
Tag des Kolloquiums: 27. Oktober 2020

Danke!

Die Arbeit an einer Dissertation ist nach meiner Erfahrung der letzten viereinhalb Jahre ein meist spannendes und lehrreiches, teils aber auch frustrierendes Unterfangen, für das man bisweilen ein nicht unerhebliches Maß an Hilfe braucht. Ich möchte deshalb an dieser Stelle all denjenigen von ganzem Herzen danken, deren Unterstützung zum erfolgreichen Abschluss meiner Promotion beigetragen hat. Allen voran ist dies mein Doktorvater Prof. Dr. Dominik Henrich, der mir die Promotion an seinem Lehrstuhl ermöglicht und dieses Projekt seit unserer ersten Idee in meinem späten Masterstudium mit seinem Rat und seiner Erfahrung begleitet hat. Weiterhin danke ich allen meinen Kollegen am Lehrstuhl: Für organisatorische und technische Unterstützung, für aufschlussreichen fachlichen Diskurs, für angenehme Kaffeepausen mit un-fachlichem Diskurs und für das Korrekturlesen, als es mit dieser Arbeit zu Ende ging. Besonderer Dank gebührt auch „meinen Studenten“, die als HiWis und mit ihren Projekt- und Abschlussarbeiten zur Umsetzung vieler meiner Ideen beigetragen haben. Überaus dankbar bin ich meiner Familie für den Rückhalt in allen Lebenslagen – besonders möchte ich abschließend meinen Eltern danken, die mir durch ihre bedingungslose Unterstützung einen Lebensweg ermöglicht haben, der mich erfüllt.

Abstract

The widespread availability of lightweight robots that may safely be operated without physical barriers to separate man and machine has paved the way to robot use in small- and medium-sized enterprises (SMEs). Given these technical foundations, the goal of advancing robots from tools to human-like teammates is a research topic that lately gains considerable attention. This thesis contributes a novel approach that particularly fosters flexible use, cost efficiency and operability by the existing workforce in line with the major requirements of partial automation in SMEs: End-users without expert knowledge on robotics are enabled to share procedural task knowledge with a robot teammate by adapting paradigms from the field of skill-based task level programming. During joint execution of previously modelled tasks, the robot is considered an equal partner of human workers – human as well as robot team members are likewise granted the authority to make dynamic, just-in-time decisions regarding the distribution of work repeatedly. This requires a high level of robot capabilities and autonomy, but therefore also allows for flexible transitions between human-robot coexistence, decoupled co-working in cooperation and close interaction in collaboration. To this end, operations from the task model are classified into categories according to their individual interaction needs and agent capabilities. An exchangeable state machine for each of these interaction categories encodes the necessary course of actions for the robot when encountering respective process steps. State machine states render the system capable of (i) understanding task progress by observing operation pre- and postconditions, (ii) executing sub-tasks itself based on a robot skill framework, (iii) delegating operations to human partners or (iv) communicating to establish mutual commitment before engaging into collaboration. Decisions in favour of an operation to go about next by following transitions in the matching state machine are made in consideration of partial workspace observability, i.e. incomplete knowledge about the state of parts and task progress: The system gets along with a lean, low-cost sensor setup only consisting of a robot-mounted eye-in-hand camera and a laser range finder to track human motion. The resulting data is fused into a human-aware world model by means of a measure for trust in stored objects. This world model enables the system to share the workspace with humans efficiently. Experiments with a simulation system that emulates dynamic human behaviour when co-working on a set of benchmark tasks show that the approach can generally speed up task execution despite these limitations in sensor use. Furthermore, preliminary human subject studies with a laboratory prototype implementation suggest that the system is promising regarding intuitive operability by non-expert users. To sum up, this thesis contributes the technical foundations, proves the feasibility of and motivates further investigations on dynamic, flexible teaming under partial workspace observability.

Zusammenfassung

Leichtbauroboter, die für den sicheren Betrieb ohne Schutzzauneinrichtungen ausgelegt sind, sind ein zentraler Wegbereiter für den Zugang kleiner und mittlerer Unternehmen (KMU) zu Robotikanwendungen. Auf dieser technischen Grundlage rückt die Frage nach der Aufgabenverteilung zwischen Menschen und Robotern in den Fokus der Forschung, sodass sie möglichst wie ein menschliches Team effektiv zusammenarbeiten können. In dieser Arbeit wird dazu ein neuer Ansatz beschrieben, der auf Flexibilität, Kosteneffizienz und Bedienbarkeit durch Endanwender abzielt und sich damit an den besonderen Anforderungen von KMU orientiert: Domänenexperten ohne besondere Robotikkenntnisse werden dazu befähigt, prozedurales Aufgabenwissen mit Roboter-Kollegen zu teilen. Dazu wird ein Verfahren vorgeschlagen, das Konzepte zur graphischen Programmierung mit Roboterfähigkeiten auf die Mensch-Roboter-Zusammenarbeit überträgt. Bei der gemeinsamen Bearbeitung einer derart spezifizierten Aufgabe werden Mensch und Roboter als Partner auf Augenhöhe betrachtet – sie entscheiden sich wiederholt für Teilaufgaben und verteilen so die Arbeit in einem dynamischen Prozess. Dies erfordert einerseits einen hohen Grad an Roboterautonomie, ermöglicht dafür aber andererseits dynamische Übergänge zwischen Mensch-Roboter-Koexistenz, weitgehend unabhängigem parallelem Arbeiten in Kooperation und eng synchronisierter Kollaboration: Dazu werden alle Operationen einer Aufgabe abhängig von der erforderlichen Interaktion und den Fähigkeiten der Teammitglieder in unterschiedliche Kategorien eingeordnet. Jeder Kategorie ist ein Zustandsautomat zugeordnet, der durch seine Transitionen die notwendigen Schritte zur koordinierten Bearbeitung entsprechender Operationen für den Roboter kodiert. Einzelne Zustände der Automaten ermöglichen es dem System dabei unter anderem den Aufgabenfortschritt durch die Beobachtung von Vor- und Nachbedingungen zu verfolgen, Operationen selbst auszuführen oder an einen Partner zu delegieren und mit dem Menschen zu kommunizieren, um mit der Ausführung einer nur gemeinsam möglichen kollaborativen Operation zu beginnen. Hinsichtlich der Entscheidungsfindung für die Auswahl einer zu bearbeitenden Operation ist das vorgeschlagene System so ausgelegt, dass dies auch unter partieller Beobachtbarkeit von Weltzustand und Aufgabenfortschritt möglich ist. So kommt der Ansatz mit einem reduzierten, kostengünstigen Satz an Sensoren aus: Die Sensordaten einer am Roboter befestigten Kamera und eines LIDAR-Sensors zur Verfolgung menschlicher Bewegungen im Arbeitsraum werden in einem Weltmodell fusioniert. Basierend auf einer heuristischen Schätzung der Wahrscheinlichkeit für menschliche Einflussnahme stellt dieses Weltmodell eine Metrik dafür bereit, wie verlässlich Daten zu länger nicht beobachteten Objekten noch sind. Das System kann somit bevorzugt Operationen auswählen, die wahrscheinlich noch verfügbare Objekte manipulieren, und so den Arbeitsraum effizient mit seinen Partnern teilen. Experimente mit einem Simulationssystem zur Emulation dynamischer menschlicher Entscheidungen mit verschiedenen Präferenzen zeigen für einen Satz von Benchmark-Aufgaben, dass die flexible Zusammenarbeit trotz eingeschränkter Sensorik die Bearbeitung der Aufgaben grundsätzlich beschleunigen kann. Ergebnisse einer Nutzerevaluation mit einer prototypischen Implementierung deuten weiterhin darauf hin, dass das System als Ganzes für Endanwender bedienbar ist. Diese Dissertation beantwortet somit die Fragestellung, inwieweit flexible Zusammenarbeit unter partieller Beobachtbarkeit des Arbeitsraums technisch machbar und für die Teilautomatisierung nutzbringend sein kann.

Contents

1. Introduction	1
1.1. Background, Motivation and Goals	1
1.2. Terms and Delimitations	6
1.3. Problem Analysis and Research Questions	10
1.4. Overview	13
2. State of the Art	15
2.1. Task Modelling	15
2.2. Decision-Making for Task Allocation	20
2.3. Coordination Mechanisms	23
2.4. Conclusions	26
3. Task Modelling for Human-Robot Teams	29
3.1. Skills for Human-Robot Teams	31
3.1.1. Domain Definition	32
3.1.2. Skill Graph Structure	35
3.1.3. Benchmark Domain	38
3.2. Shared Task Model Generation	43
3.2.1. Graphical Modelling of Precedence Graphs	43
3.2.2. Annotation with Operation Pre- and Postconditions	46
3.3. Task Execution Principle	47
3.4. Conclusions	48
4. Human-Aware World Modelling for Task Allocation	51
4.1. World Model Definition and Maintenance	52
4.2. World Model Ageing	56
4.2.1. Human Workspace Model	57
4.2.2. Interaction Indicators	58
4.2.3. Trustworthiness of Data	61
4.3. Metrics for Task Allocation	63
4.4. Conclusions	64

5. Coordination of Flexible Human-Robot Teams	67
5.1. Team Mental Model	68
5.1.1. Agent Capability Model	70
5.1.2. Interaction Categories	72
5.1.3. Flexible Communication Patterns	74
5.1.4. Preemptive State Machines	77
5.2. System Architecture	80
5.3. Dynamic Task Sharing	82
5.3.1. Decision-Making Strategies	83
5.3.2. Task Advancement	85
5.3.3. Knowledge Update	86
5.4. Conclusions	89
6. Evaluation	93
6.1. Subjective Evaluation	94
6.1.1. Hardware Prototype	94
6.1.2. Results	97
6.2. Objective Evaluation	99
6.2.1. Benchmark Tasks	99
6.2.2. Simulation System	101
6.2.3. Parametrisation	102
6.2.4. Evaluation Metrics and Reference Data	103
6.2.5. Results	105
6.3. Conclusions	112
7. Conclusions	115
7.1. Summary and Discussion	115
7.2. Future Work	119
A. Complementary Evaluation Data and Parametrisation Details	123
List of Tables	127
List of Figures	129
Bibliography	131

CHAPTER 1

Introduction

1.1. Background, Motivation and Goals	1
1.2. Terms and Delimitations	6
1.3. Problem Analysis and Research Questions	10
1.4. Overview	13

WITHIN the last years, robots have started to evolve from potentially dangerous, inflexible automation tools towards teammates in manual production processes. Lightweight robots are broadly available. They enable coexistence with humans in a shared workspace – fences as used in traditional automation are no longer necessary to ensure worker safety. Such robots are the enablers of automation in small and medium sized enterprises (SMEs) and workshops when combined with novel approaches to intuitive end-user robot programming. There is, however, still a lack of cognitive robot capabilities to make use of the positive effects of symbiotic, human-like teaming beyond mere coexistence in these production environments. Against this background, this thesis addresses human-robot teaming with a focus on flexibility as motivated in Section 1.1. This form of teaming is put into the overall context of human-robot interaction in Section 1.2. An in-depth problem analysis in Section 1.3 leads to the research questions that this work seeks to investigate.

1.1. Background, Motivation and Goals

The field of industrial robotics is currently undergoing an evolution. Robots were traditionally highly specialized tools for fully automated mass production. They were programmed statically for long-term use in a single, fixed task. Fences guaranteed the safety of workers on the shop floor by introducing a strict separation between man and machine. This inflexible mode of use has prevented SMEs from broadly utilising robots so far – high acquisition costs, space requirements caused by safety fences and frequent expenses for expertise in robot programming have hampered automation of small batch production [93, 99]. Regarding these issues, the introduction and broad availability of



Figure 1.1.: Lightweight robots like e.g. the Franka Emika Panda (left) or the KUKA LWR 4+ (right) arms are compact manipulators designed for use in cage-free applications.

lightweight robots can be seen as a turning point. Meanwhile, lightweight robots as shown in Figure 1.1 are available from all major robot manufacturers (see e.g. [124] for a detailed listing). They are characterised by their compact design and reasonable pricing on the one hand [13]. On the other hand, lightweight robots are often equipped with built-in safety technologies [115]. Different approaches to measure external forces and torques, tactile sensor systems to detect contact, and insights from the field of soft robotics can prevent or at least bound the impact of collisions with workers. These mechanisms are the key to fence-less coexistence of humans and robots in manufacturing: Partly automated applications can be designed in line with the relevant safety standards EN ISO 10218-1/2 [34, 35] and ISO/TS 15066 [56] when safe robots are integrated with equally safe actuators (e.g. grippers as developed in the SCHUNK Co-act technology program¹) and non-hazardous workpieces. The resulting hybrid workplace needs to undergo a risk assessment procedure to prove compliance with the maximum acceptable contact situations, forces and pressures defined in ISO/TS 15066 [124]. Even direct physical interaction with robots can then be certified in line with European regulations and laws on safety and health at work.

Programming is the activity that takes up most of the time in industrial human-robot interaction scenarios [124]. It has been identified as one of the main barriers to automation in SMEs [99, 93]. Alongside with the reduced capital cost and enhanced flexibility offered by fence-less lightweight robots, automation in SMEs is thus supported by research on intuitive end-user programming techniques. Especially Programming by Demonstration and Task-Level Programming with skills have gained considerable attention in this field [117]. Both paradigms are often combined with graphical user interfaces (e.g. [117, 96, 102]). Such interfaces are meanwhile also commercially available and open robot programming to shop floor workers, e.g. through Franka Emika’s Desk interface for their Panda arm² or Intera Studio³ offered for Sawyer by Rethink Robotics.

In summary, suitable hardware, software and legal frameworks are available to use robots in manufacturing of frequently changing products with an increasing range of variants. Recent data indicates that many SMEs have already started to implement these technologies or are planning to do so within the next few years [63]. However, robots are currently mostly used as tools that merely coexist with humans rather than taking the role of a co-worker [13]. Most concrete applications do thus not make use of

¹https://schunk.com/de_en/co-act/ (date accessed: 2020-05-25)

²<https://www.franka.de/capability> (date accessed: 2020-05-25)

³<https://www.rethinkrobotics.com/intera> (date accessed: 2020-05-25)

the positive effects of true teamwork that are emphasized by an ever-increasing number of publications on human-robot cooperation and collaboration [2]: Humans and robots have different capabilities. Manipulators excel in strength, endurance and precision, but they are currently rather inefficient when it comes to dexterous manipulation [115]. This weakness can be compensated for by humans who moreover possess the cognitive skills and intuition to adapt quickly even in unexpected situations. These individual capabilities need to be combined in a synergistic way to create symbiotic assembly systems of the future [37]. Sharing a task can obviously raise productivity when team members work in parallel. In addition to reduced makespans, the introduction of human-robot teams can improve working conditions. Physical stress can systematically be shifted from shop floor workers to machines by considering ergonomics in the design of hybrid assembly systems [97]. All in all, worker support and enhanced productivity through human-robot teams in SMEs may be the answer to the reduced workforce and ageing population induced by future demographic changes [8, 115]. Against this background, this thesis is directed towards advancing lightweight robots from tools to human-like team members by adopting the aforementioned developments that brought them to SMEs. Therefore, the design of the proposed human-robot teaming framework is guided by the following two main goals:

- G1 Maximize Flexibility:** The system must support fast and flexible integration of new tasks by accepting shared task models as an input. Humans and robots must be seen as peers that share work dynamically during the on-line execution phase.
- G2 Minimize System Costs:** The overall costs for acquiring the system and operating it should be kept low by relying on a lean hardware setup within a software toolchain that the existing workforce can handle.

A system will not be suitable for production of small lot sizes and ever-changing products without flexibility regarding new tasks (G1). Observations from human teaming have shown that a *shared mental model*, i.e. a similar understanding of the task among team members, is needed for effective teamwork [82, 74]. Human mental models allow to describe and predict the behaviour of systems [59] and are rather complex. Mathieu et al. have identified task models and team models as the two main content domains of mental model information [82]: The *task mental model* mainly describes resources and procedural knowledge on a task. Demanding a *shared* task (mental) model that all team members know and understand as a system input is thus not only a matter of flexibility but also ensures the common ground that humans and robots need to function as a team.

By contrast, the *team mental model* covers one's understanding of teammates' roles, skills, preferences, communication channels and interaction patterns etc. For a robot system, this part of the mental model strongly relates to how the system implements planning of actions and communication. This dissertation seeks to realise a robot teammate with a team mental model that supports the following mode of teaming: We know from experience that human teams can act in a highly dynamic and flexible way. In particular, effective teams often make the decisions to distribute work on the fly rather than strictly following a pre-planned, fixed schedule [112]. According to Tracht

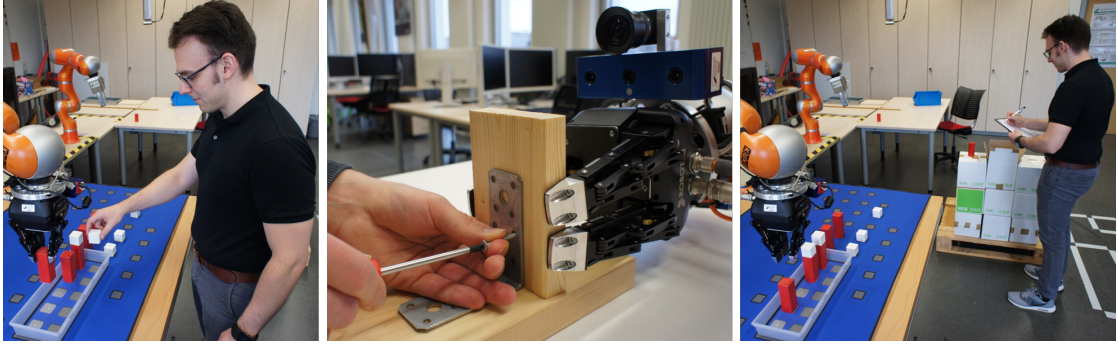


Figure 1.2.: This thesis targets a human-robot teaming framework that enables efficient, decoupled parallel working (e.g. cooperative palletizing, left), that can enter close interaction for collaborative action when needed (e.g. assembly steps requiring a helping hand, centre) and that allows human agents to handle more urgent tasks temporarily (e.g. a delivery of goods, right).

et al., enabling humans to influence the working process can impact worker satisfaction positively [122]. Excluding humans from the decision making process can even result in decreased human situational awareness [40] – despite all built-in safety measures of lightweight robots, critical situations may still require quick operation of the emergency stopping mechanisms demanded by ISO/TS 15066 [56]. Shared authority can hence also contribute to safety by keeping workers alert at any time. Furthermore, recent results from the field of social robotics confirm that humans prefer proactive robots to passive assistance systems that strictly follow human decisions and commands [9]. In conclusion, humans and robots should be equally responsible for initiating and carrying out parts of the task. Both should equally possess decision making authority on their own rather than strictly depending on their partner. Ideally, workers will then make decisions contributing to capability-based division of work after gathering experience with the system. Besides these general findings, mimicking human teams with a flat structure instead of a hierarchical organization has beneficial practical implications on the flexibility of human-robot teaming: Working in a decoupled, parallel way (Figure 1.2, left) as far as possible does not only result in reduced makespans and enhanced productivity – it also lets shop floor workers a choice: They can intentionally leave obnoxious, repetitive sub-tasks to the robot and may even temporarily walk away from the workbench while the robot keeps working. Workers can this way handle interruptions and re-engage in the task later, e.g. when replenishing resources or receiving a delivery of goods (Figure 1.2, right). Nevertheless, the team must be able to engage into closer, potentially physical interaction when needed. Complex actions as e.g. two-handed manipulation requiring an additional helping hand (Figure 1.2, centre) that neither teammate alone is capable of are otherwise intractable.

The demand for maximum flexibility is intertwined with the system costs. On the one hand, the cost factor and a lack of expertise are major concerns regarding the introduction of robots in SMEs [99, 63]. On the other hand, robot teammates need an extensive set of capabilities [72], especially when they are intended to work with humans dynamically by

making decisions in ever-changing environments. These capabilities of cognitive robots come along with a need for additional hardware, e.g. for perception and communication. The second goal seeks to address these issues by firstly demanding a lean hardware setup. This can be achieved by reducing the number of used sensors to a minimum (e.g. one camera mounted near the robot hand to partially observe the workspace) and by reusing hardware that is either way available in SmartFactory environments (e.g. workers' smartphones). However, G2 does not only relate to hardware investment costs – it also aims to reduce the need for external expertise and additional personnel expenditure. SMEs typically employ domain experts that are skilled craftsmen and product engineers [99]. That is why they do not necessarily have expertise in robotics. Another key motivation of this dissertation is to enable these domain experts to operate the full system. This means that they must not only be able to coordinate work with their teammate during production intuitively – procedural know-how must furthermore be preserved by fostering knowledge transfer towards the robot with regard to task mental models. This can be achieved by applying insights from prior art on intuitive robot programming to flexible human-robot teaming. Moreover, the existing workforce could ideally install the system, put it into operation and carry out maintenance tasks (e.g. (re-)calibration of sensors after moving the robot on the shop floor).

Against the background of the aforementioned goals G1 and G2, the central working hypothesis of this dissertation can therefore be formulated as follows:

Flexible human-robot teaming by dynamic task sharing can be beneficial for production processes, even if a hardware setup with only a few sensors is used. Users without expert knowledge in robotics can operate the system and can intuitively share their procedural know-how before working with a robot teammate.

The need for investigating this hypothesis is substantiated by a short review of related work: Approaches to human-robot teaming with applications to manufacturing mostly focus schedules for an ideal, capability-based distribution of work (e.g. [86, 78, 84, 58, 16]). By searching for optimality e.g. regarding ergonomic aspects, these methods lead to schedules that are fixed and do not enable flexibility by dynamic just-in-time decisions. More dynamic modes of task sharing have been considered in literature yet (e.g. [60, 30, 103, 44, 89, 112]), but they predominantly require precise knowledge of the task progress at any point in time – this sort of full observability can only be achieved by monitoring human actions or the state of the workspace with extensive sensor setups. Recent approaches to graphical robot programming rely on skill frameworks to support end-user programming in industrial settings, but they result in commands for the robot only and so they are not suitable to generate shared task models for multiple agents (e.g. [117, 4]). Thus, there is a gap in knowledge regarding dynamic task sharing with limited sensor use in combination with task models that are based on the established task-level programming paradigm for manufacturing scenarios. This thesis seeks to contribute to prior knowledge by investigating the prospects and limitations of flexible teaming in line with these considerations. To this end, a novel approach to dynamic task sharing is evaluated with a particular focus on the impact of partial workspace observability.

1.2. Terms and Delimitations

Section 1.1 has motivated a flexible form of teaming that targets cost-efficient hybrid assembly stations where humans and robots interact dynamically to achieve a shared goal. Since human-robot interaction is a rather wide field of research [41] it is necessary to classify flexible teaming carefully and delimit the approach to problems that are not addressed by this thesis. To this end, the following terms and definitions are crucial:

Task In the context of this work, humans and robots pursue the shared goal of completing a shared task together. A *task* describes a complex process, as e.g. assembling a product from several parts or packaging different goods onto a pallet.

Operation Tasks are composed of several operations. An *operation* is a self-contained unit that realises a part of the task, e.g. by mating one part with an assembly.

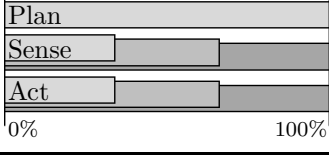
Task Allocation The process of selecting a subset of agents (human, robot or both) for execution of some operation is called *task allocation*.

The term *dynamic task allocation* will be used for just-in-time decisions that an agent makes to choose his or her next operation from a given task model.

A classification of the approach using the established taxonomy of Yanco and Drury [127] is listed in Table 1.1. The task type and robot morphology are prescribed by the target domain of manufacturing with lightweight robots in SMEs. The task criticality is medium: Success of the shared task is not entirely irrelevant, as failures in production cause costs and lower productivity. However, teaming in manufacturing is not as critical as e.g. success in search and rescue scenarios, in which failure may affect the life of humans [127]. One robot will interact with possibly several shop floor workers as peers.

A high robot autonomy level is the key enabler of peer-to-peer interaction [41]. The taxonomy of Yanco and Drury specifies autonomy to be ‘the percentage of time that the robot is carrying out its task on its own’ without human intervention in the control process [127]. This measure depends on agent capabilities and the concrete task if this definition is directly applied to our case: If some task contains a sufficient amount of operations that the robot is capable of, then 100% of autonomy within the time needed for task completion is possible. By contrast, a task that predominantly involves dexterous assembly steps (Figure 1.2, centre) results in low autonomy. A general statement about the degree of autonomy regarding this definition is thus infeasible. A more differentiated classification may be taken by additionally considering autonomy sub-categories for *planning*, *sensing* and *acting* [11]. As described above, the autonomy level for acting in the sense of performing operations within a task scales depending on the relation between capabilities and operations – it may vary for different task structures (different shades of grey in Table 1.1). The same is true for sensing: Some task may e.g. involve operations to shelve objects out of the reach of a manipulator. Then, the stationary robot is not capable of verifying the completion of a fraction of the operations without help. The idea of tasks that require a combination of complementary agent capabilities is reflected by equally granting human agents a limited degree of sensing and acting autonomy,

Table 1.1.: Classification of the interaction targeted by this work regarding relevant categories of the taxonomy by Yanco and Drury [127] with additional autonomy sub-categories according to Beer et al. [11]

Category	Value
Task Type	manufacturing
Robot Morphology	functional
Task Criticality	medium
Human-Robot Ratio	several humans, one robot
Interaction Roles	teammates
Autonomy	

e.g. due to limited physical strength or workspace areas that only the robot can access. However, human domain experts are usually aware of these circumstances at any time due to their experience and cognitive skills. This work targets a robot teammate that is similarly capable of understanding the situation and reacting by planning autonomously during 100 % of the task execution time. These reasoning capabilities help to bridge the gap between individual agent skills by bidirectional communication, e.g. by asking for information or by requesting help in performing operations. Bearing in mind the idea of dynamic task allocation, this work strives for an approach to *teamwork-centred autonomy* that ‘adopts the stance that the process of understanding (...) and task execution are necessarily incremental, subject to negotiation, and forever tentative’ [20].

The original taxonomy of Yanco and Drury contains further categories regarding physical proximity and sharing of time and space [127]. These categories are not contained in Table 1.1 due to the fact that they are covered by the terms ‘coexistence’, ‘cooperation’ and ‘collaboration’. These terms are frequently used in recent publications [2] and commonly applied in the field of industrial co-working [13, 12]. Using them for classification is certainly more suitable in the context of this work. Unfortunately, the meaning of coexistence, cooperation and collaboration is not defined consistently in literature [1]. The teaming scenarios targeted in this thesis are best matched by definitions inspired by those of Aaltonen et al. [1], Bender et al. [13] and Behrens et al. [12]. All terms are based on the assumption that human and robot at least share a part of their workspace:

Human-Robot Coexistence: Human and robot *coexist* when they share parts of their workspace without fences, but do not work on the same task.

Human-Robot Cooperation: Human and robot *cooperate* when they work on the same task by carrying out different operations involving different objects simultaneously.

Human-Robot Collaboration: Human and robot *collaborate* when they work on the same operation contained in the same task simultaneously. Physical contact, possibly transmitted via an object to be handled jointly, is allowed and intended.

The meaning of these definitions is visualized by Figure 1.3. One can see that each of the teaming modes covers one of the motivational examples shown in Figure 1.2. We can

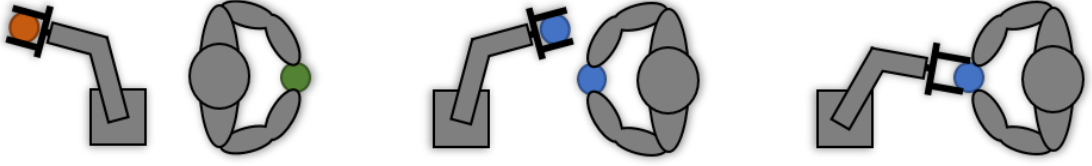


Figure 1.3.: When humans and robots *coexist* (left), they work on different tasks that are represented by different objects (orange/green). *Cooperation* means that agents work on separate parts contributing to the same task in parallel (centre). The term *collaboration* additionally involves temporal synchronization and physical interaction within an operation (right).

assume that coexistence mainly refers to the ability of humans to switch to another task as stationary manipulators are bound to their limited workspace. The robot is intended to still stay productive by proceeding with the shared task even if its partner has left. The notion of flexible teaming can then be defined more formally as follows:

Flexible Teaming A *flexible team* works on one or more tasks without relying on a fixed, precomputed schedule and can hence dynamically switch between cooperation, collaboration and coexistence at any time (Figure 1.4, left).

Task allocation decisions of humans and robots are not sufficient for meaningful teaming. Coordination, defined as ‘the act of managing interdependencies between activities performed to achieve a goal’ [79], is additionally required to react to the actions of one’s peers – this means in particular that dynamic task allocation must reflect task progress in terms of operations completed by teammates and leads to the notion of task sharing:

Dynamic Task Sharing The term *dynamic task sharing* is defined to denote a process of coordinated human and robot dynamic task allocation decisions.

Coming from the three teaming modes and potential transitions among them, coordination can be regarded on different levels of granularity (Figure 1.4, right): Collaboration needs the most fine-grained coordination. To this end, the *operation level* covers the tightly coupled interaction needed for assembly steps that involve more than one agent synchronously. While this thesis incorporates basic means of operation-level coordination, this level can generally be seen as the interface to numerous approaches to physical human-robot interaction (pHRI) and shared control, e.g. for collaborative lifting or other manipulations with physical contact [105]. The team must furthermore coordinate on the *task level* during cooperation. Task level coordination encapsulates task allocation, i.e. agents share responsibility for parts of a complex task on this level. In contrast to the operation level, team members can act rather independently and are not strictly tied to each other – their actions are only constrained by a loose temporal coupling originating from potential task-related constraints on the ordering of operations and availability of resources. Coexistence does not need any task-related coordination, as agents work on differing tasks in this mode. However, the transition between coexistence, cooperation and collaboration demands for additional coordination on the *teaming mode level*. Let

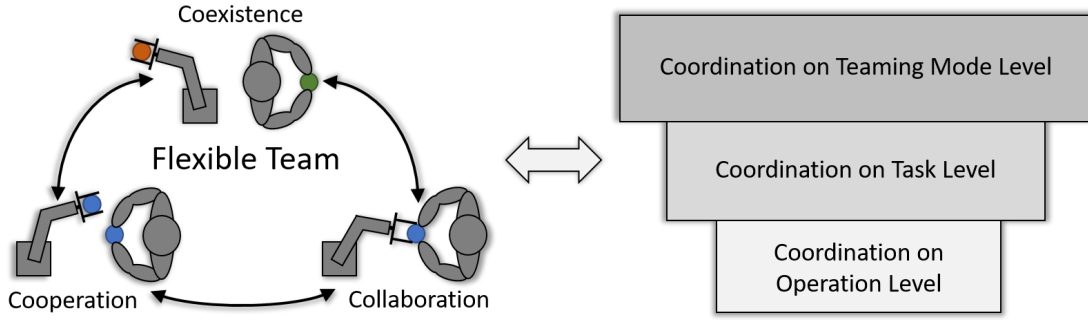


Figure 1.4.: Flexible teams can switch between coexistence, cooperation and collaboration dynamically (left) by coordinating actions on different levels of granularity (right).

the robot e.g. decide to carry out an operation that is only feasible in collaboration, while the human partner is absent from the workbench. Then, the communication to ask the human to return and support the operation falls within the scope of teaming level coordination. The earlier mentioned autonomy in robot planning must reflect all levels of coordination for flexible teaming.

We know that solving the problem of autonomous reasoning and planning in general is still one of the greatest challenges in robotics [128]. The scope of this thesis is delimited by the following assumptions: A shared goal is encoded by a symbolic task model describing the task and all resources needed to complete it. This model is shared between humans and the robot, i.e. it is fully known to all agents. The problem of recognizing objects in sensor data, especially in images, is well studied (cf. e.g. [77, 129]). Relying on this body of work, it can furthermore be assumed that a robot is able to recognize the state of any resources (e.g. the type and location of objects) as soon as they occur in the data produced by any sensor of the system. Focussing on limited sensor setups, the notion of partial workspace observability can be clarified against this background:

Partial Workspace Observability Perception under *partial workspace observability* means that only a fraction of all objects in the workspace can be observed by the robot at a given point in time.

Consequently, planning and reasoning in the context of this work means generating actions to perceive a subset of the objects in the workspace or to contribute to task progress correctly. We furthermore assume *cooperative workers* who are skilled within the domain and who will always perform correct actions according to the task model without causing errors in the production process.

All in all, the research directions and targeted areas of contribution can be summarised as follows: We have seen so far that task allocation and coordination are central prerequisites for flexible human-robot teaming on the one hand. On the other hand, cost-efficient end-user programming and system operation are key enablers of robotics in SMEs. The principal focus of this work lies on a formal framework that integrates prior findings on user-friendly modelling with dynamic allocation and coordination of

tasks – we will therefore encounter these central terms again when considering related work in depth in Chapter 2. The scope of this thesis is bounded by conceptual interfaces to integrate existing approaches to explicit human-robot communication and physical interaction into the task allocation and coordination process. The issue of safe coexistence is extensively covered by a vast body of literature (cf. for example the surveys of Lasota et al. [69] and Halme et al. [43]). Intrinsic worker safety (e.g. through force and power limiting according to ISO/TS 15066 [56]) is thus assumed as given and not addressed explicitly. Aside from task-related communication, social aspects of human-robot interaction (e.g. expressing emotions, maintaining relationships or exhibiting a distinctive character [38]) are out of scope with regard to the target industrial application. Making individual decisions on task allocation and coordinating them within a team means considering one’s partners’ intentions and plans implicitly. The survey of Bauer et al. [10] lists numerous channels that can convey intention – from these channels, only explicit communication and observation of object usage are used by the proposed task sharing approach. Especially methods for explicit recognition of gestures or actions by tracking of human motions are not used due to the focus on partial workspace observability.

1.3. Problem Analysis and Research Questions

Coming from the high-level goals set in Section 1.1, we can now look at the technical aspects of flexible teaming within the scope delimited by Section 1.2. These are directly interrelated with the research questions that this thesis tries to answer. The first question addresses the issue of establishing a task-related shared mental model. Domain experts already possess the process knowledge on tasks to be partly automated. The problem can thus be reduced to transferring task models from human to robot agents. A suitable task model must therefore meet several requirements: It needs to be legible and comprehensible for humans so that end-users can manage task modelling. But the model must still encode sufficient information for the robot to perceive and understand task progress. It must additionally contain information on whether operations may be executed in parallel for task sharing in the cooperation mode. Graphical robot programming environments have recently been in the focus of research and commercial applications – it is therefore reasonable to explore if the advantages of this paradigm can be transferred to flexible teaming. To sum up, these requirements lead to the following first research question:

Q1 To what extent can end-users without knowledge on robotics use a graphical interface to create task models that can be shared with robots for flexible teaming?

Given the shared task model, the second research question targets the issue of dynamic task allocation. Goal G2 in mind, this should be achieved with as few as possible inexpensive sensors to reduce acquisition costs, expenditures for installation and efforts for calibration. The system will therefore mainly rely on an eye-in-hand RGB-D camera system that is mounted near the robot tool centre point. Installing a single robot-mounted camera is far less invasive compared to a complex multi-camera system that can oversee the whole workspace. Suitable cameras with a compact form factor are available at low

cost (e.g. Intel’s Real Sense D435 depth camera⁴). Aside from the cost factor, using a robot-mounted camera system has an application-specific motivation: Lightweight robots only provide a limited payload. Target application scenarios for human-robot teaming thus incorporate manipulation of parts that are relatively small compared to the agents moving within the workspace. As a result, a high level of occlusion may be expected when using cameras in fixed positions. This issue can be avoided by enabling the robot to move the camera and inspect parts actively when needed (Figure 1.5). However, these benefits of partial workspace observation raise another challenge: Humans may arbitrarily change the state of objects that are currently not in sight of the camera – the knowledge that the robot has previously gathered can thereby be invalidated, causing it to make decisions based on outdated information. In consequence, the robot may e.g. decide to handle an operation that has already been done by a human worker in the meantime, resulting in inefficient robot behaviour or even erroneous task execution. Providing the robot with knowledge on human habits, e.g. by demonstrating the workers’ preferred ways to execute the task, would certainly support decision-making. It is on the other hand time-consuming and costly due to down-times in the training phase. This thesis therefore aims to contribute to prior knowledge by answering the following question:

Q2 How far can a robot system make meaningful dynamic task allocation decisions under partial workspace observability without extensive prior training?

In the attempt to explore the boundaries of dynamic task allocation under partial workspace observability, the sensor set used in this work is complemented by a low-cost 2D LIDAR range sensor (e.g. Slamtec’s RPLIDAR A2⁵). Such sensors can only provide tracking information about the 2D position of humans in the working area when mounted below the workbench on the one hand. On the other hand, data from a single LIDAR sensor can cover the whole operational space of the team – this would otherwise e.g. require several, labour-intensively calibrated cameras. The resulting overall system setup is depicted in Figure 1.5. This setup enables investigating the effects of gradually adding knowledge about human motion to the system. Still, it is a reasonable trade-off that avoids turning to full observability as e.g. enabled by camera-based full human body pose tracking or even more expensive and intrusive marker-based hardware setups [107, 76] that are not in line with the goal of a lean hardware setup.

Even when provided with additional, but still partial information on human motion in the workspace, imperfect decisions may still occur. The system must then potentially cope with failed operations, e.g. by retrying them later or by asking the human partner to clarify the situation. Communication is furthermore necessary to implement the different levels of coordination introduced in Section 1.2. Otherwise, the robot e.g. will not be able to call its partner for help with a collaborative operation that it cannot handle alone. However, the partner will not necessarily immediately react to a request in a flexible setup – a team mental model (Section 1.1) is thus needed as a protocol for the course of action in such situations. With this model, the robot system can decide after each task

⁴<https://www.intelrealsense.com/depth-camera-d435/> (date accessed: 2020-05-26)

⁵<https://www.slamtec.com/en/Lidar/A2> (date accessed: 2020-05-26)

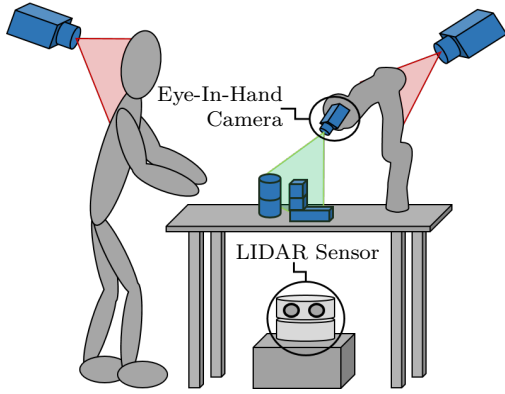


Figure 1.5.: The limited hardware setup consists of an eye-in-hand camera mounted near the robot tool centre point and a LIDAR range scanner below the workbench. Active inspection of parts (green) avoids occlusions as emerging from cameras in fixed positions (red).

allocation decision whether to (re-)try an operation, if communication is necessary or how to handle a lack of response etc. The demand for flexible teaming comes with a need for a communication channel that is equally flexible. In particular, the team mental model formulation for the robot must realise a mode of communication that is non-blocking and stoppable to the greatest possible extent. Only then will the system stay capable of acting in situations when the human is occupied otherwise and does not answer to requests – blocking communication would here prevent the system from handling other operations in the meantime. The following research question reflects these requirements:

Q3 How can dynamic task allocation be combined with flexible communication and a team mental model to integrate coexistence, cooperation and collaboration?

Although verbal and gesture-based human-robot communication are well studied [83], short messages via handheld devices are more suitable for a prototype implementation of the proposed approach. Voice- and gesture-based systems are still seen sceptically by industry professionals [63]. Moreover, speech recognition performance without intrusive headsets depends strongly on noise, e.g. due to actuator motion [92]. By contrast, short messages do not need any hardware aside from already omnipresent smartphones. They moreover reach off-workplace partners that are out of earshot in the coexistence mode.

The above questions focus conceptual feasibility and limitations of dynamic task sharing under partial workspace observability. A final question seeks to provide a more general view on the prospects of flexible teaming with a limited sensor setup for manufacturing. Section 1.1 has introduced shop floor workers as the end-users of the proposed system – these domain experts do not possess expert knowledge on robotics, but they should still be enabled to operate the system without relying on external expertise. Even if the system can be run cost-efficiently by the existing workforce, it must still provide a benefit compared to fully manual work to render human-robot teaming reasonable:

Q4 To what extent is the proposed method beneficial for shared task execution? How does the flexible approach compare to static methods for task sharing?

This question particularly targets the influence of decision-making under partial workspace observability on teaming performance. An overview of the approach used to investigate the above questions is given in the next Section 1.4.

1.4. Overview

The parts of this thesis still to come are structured as follows: Chapter 2 reviews related work to build upon. This chapter also clarifies the gap in knowledge more precisely. An end-user task modelling concept is introduced in Chapter 3. Starting from a task-level robot skill framework, complex tasks are composed using a graphical programming tool. Each task is represented by an assembly precedence graph. Operations are annotated with automatically generated pre- and postconditions for perception of task progress.

A human-aware world model is the second key component of the proposed human-robot teaming method (Chapter 4). This world model stores symbolic data about parts in the workspace. It is continuously updated from partial views of the workspace as provided by the eye-in-hand camera system. Partial observability is handled by a trust measure to judge reliability of stored objects over time. This measure is incrementally calculated based on heuristics for the likelihood of parts being manipulated by humans.

Chapter 5 brings things together by describing the human-robot interaction algorithms. The human-aware world model provides guidance for the robot to choose an operation from the task model: Dynamic task allocation decisions are taken in favour of operations that are likely to succeed due to high trust measures of involved parts. Each operation is assigned to a component team mental model in consideration of agent capabilities. These component mental models encode workflows that are necessary to coordinate respective actions. For instance, an operation that the robot cannot perform requires communicating this fact to its partner and monitoring postconditions afterwards. By contrast, operations that both agents are capable of may be carried out after validating preconditions, collaborative operations should only be issued after establishing mutual commitment to engage into close interaction etc. Mental models are therefore formulated as state machines that combine abstract robot activities for (i) active inspection of parts with the camera, (ii) execution of an operation from the task model, and (iii) explicit communication with the human. After choosing an operation to work on, necessary activities towards completing this operation are issued by following transitions in the corresponding mental model state machine.

Two sorts of experiments support the evaluation of this approach (Chapter 6): A hardware prototype for pick-and-place tasks and basic assembly steps is used to test the system with human subjects. This enables investigating user experience with different stages of a software toolchain for flexible teaming. By contrast, a simulation system allows for an in-depth analysis of teaming performance for different tasks and system parameter sets based on partly randomised simulation of human participation. Finally, Chapter 7 summarises the approach, discusses the evaluation results and points out future research directions.

CHAPTER 2

State of the Art

2.1. Task Modelling	15
2.2. Decision-Making for Task Allocation	20
2.3. Coordination Mechanisms	23
2.4. Conclusions	26

SHARING a task model, being able to make decisions on task allocation, and coordinating these decisions with one's peers are crucial for flexible teaming (Chapter 1). Therefore, this chapter reviews related work on human-robot task sharing from each of these three perspectives: Section 2.1 gives an overview of task models that are used in the context of human-robot teaming. The problem of making decisions on allocating parts of a task to either human or robot is considered in Section 2.2. Existing technical approaches to team coordination are outlined in Section 2.3. Section 2.4 summarises the lessons learned and clarifies the gap in knowledge that this thesis seeks to address.

2.1. Task Modelling

Shared task models are an integral part of human-robot interaction approaches that aim to achieve a common goal as a team. Hence, related work is categorized regarding usage of such models as an input hereinafter. Properties of individual task model types are summarised in Table 2.1. The properties taken into account result from the requirements for applicability to graphical end-user programming and flexible teaming as identified in Section 1.3. In consequence, only task models with an explicit structure that can be visualized graphically are regarded here – this excludes works from the below considerations in which the task is defined implicitly by specifying the goal state as a CAD product model [14, 84, 61], as an abstract goal [111, 33, 85], or by predicate target values [24, 48] for symbolic planning. Besides explicit encoding of parallel task execution, complexity is rated on a scale from ‘low’ to ‘very high’. Models of low complexity express exactly one possible way to perform a task. By contrast, medium complexity refers to representations that can capture more than one possibility. If a model covers knowledge

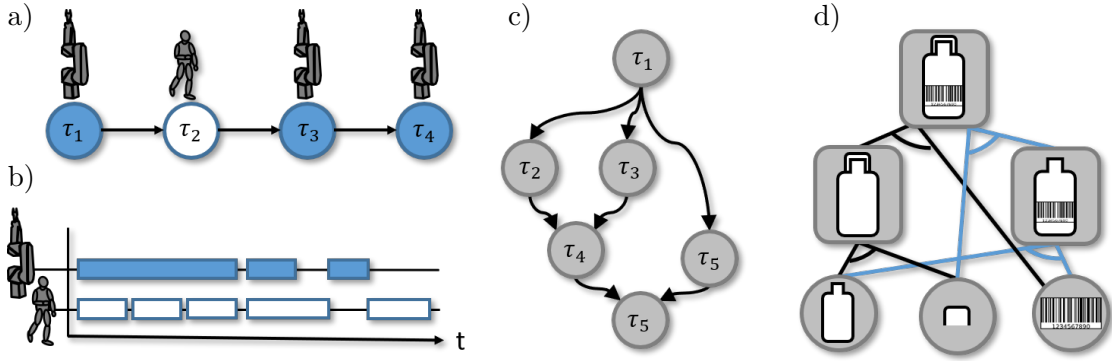


Figure 2.1.: Operation sequences (a), schedules (b), precedence graphs (c) and hierarchical decompositions (e.g. AND/OR Trees) (d) are common input task models to human-robot teaming approaches in literature.

beyond the task structure in terms of operations and constraints on their ordering (e.g. expert information on the duration of operations or on robot perception capabilities), then complexity is rated high. Task models that are based on additional information that is particularly complicated to obtain (e.g. probability distributions on the outcomes of actions) are classified as very highly complex. The last column of Table 2.1 indicates, if and how user-friendly modelling is addressed in respective publications.

Operation Sequences are a basic form of task modelling. Against the background of robot use in industrial settings, the definition of operation sequences is chosen in line with the notion of assembly sequences: Following Homem de Mello and Sanderson [54], a totally ordered set $\tau_1, \tau_2, \dots, \tau_i, \dots, \tau_N$ ($N \in \mathbb{N}$) of assembly operations τ_i is called an *assembly sequence* if and only if τ_N results in the final product, and if each τ_i uses only single parts or sub-assemblies resulting from τ_k with $k < i$ (Figure 2.1a). While the term ‘assembly operation’ is mainly used for physical mating of parts, the more general definition as a self-contained part of some task (Section 1.2) is meant hereinafter. Operation sequences provide only a low level of information content. They are therefore easily comprehensible and legible for humans. In the context of human-robot teaming, approaches to learn sequences from observation of humans have been proposed [67, 64]. Kimura et al. have shown, how robots can issue supportive actions while humans process a sequential task [64]. In the framework proposed by Lallée et al., a robot may take the role of either of two agents that were involved in the demonstration [67]. In general, one operation sequence does not encode parallelism. However, parallel execution can emerge when individual operation sequences for both human and robot are combined with operation start and end times to form a *schedule* (Figure 2.1b). Schedules are not directly used as an input task model in literature – they are thus not contained in Table 2.1. Still, the notion of schedules is important as they are a common output of task allocation methods starting from one of the more complex task models presented below.

In contrast to operation sequences, **Precedence Relations** are a fundamental assembly representation that directly encodes parallelism [91]. This model specifies tasks as a set $T = \{\tau_1, \tau_2, \dots, \tau_{|T|}\}$ of $|T|$ operations, and a partial order $<_T$ on the elements

of T . A *precedence relation* $\tau_i <_T \tau_j$ ($\tau_i, \tau_j \in T, i \neq j$) means that operation τ_i must be completed before τ_j , e.g. due to geometrical constraints when mating parts. Precedence relations can be visualized as **Precedence Graphs** (Figure 2.1c) according to the following definition of Niu et al.:

"A *precedence graph* is a directed graph, whose nodes represent the assembly operations on parts and whose arrows represent the precedence relations between operations." [91]

E.g., τ_4 may only be executed after finishing τ_2 and τ_3 in Figure 2.1c. Precedence graphs combine many feasible operation sequences into one single representation. The information content is therefore higher compared to a single sequence. This is accompanied by reduced legibility due to increased complexity. Given a precedence graph, an optimal assembly sequence regarding certain operational criteria (e.g. minimum cycle time) is determined by *assembly sequencing* [57]. Related optimisation and scheduling approaches have been extended to create schedules that take human and robot individual capabilities into account [86, 78, 90, 97, 16]. These works do not explicitly address modelling of input precedence relations [86, 78, 90, 16] or require expertise in standardized hierarchical task analysis [97]. There are, however, methods to generate precedence relations automatically from high-level product information (e.g. [91]) which can replace modelling by end-users. Recent work done by Mateus et al. includes this feature [81] but relies on decisions of expert workplace designers in subsequent steps of the planning process.

Temporal Constraint Networks are similar to precedence graphs as they also impose ordering constraints onto a set of operations. Their information content is rated higher because operations τ_i are furthermore characterised by their start and end points in time $t_{\text{start},i}$ and $t_{\text{end},i}$. Let X denominate the set composed of all these points in time. Then, temporal constraints a_{xy} and b_{xy} on the duration of operations or the overall task are formulated by demanding that $x - y \in [a_{xy}, b_{xy}]$, $x, y \in X$ [112]. Solving the temporal constraint problem defined by such networks means searching a value for each point in time, so that all constraints are satisfied. The Chaski framework of Shah et al. applies this model to human-robot teaming by considering individual bounds on operation durations for each agent in a team [112]. The scheduler of Wilcox et al. considers human preferences regarding the workflow as an additional model component [40, 126]. Levine and Williams have proposed a similar approach that integrates alternatives and choices into a single task model [73]. Specifying temporal constraints and preferences requires expert knowledge, e.g. on human worktime analysis and robot motion duration – this hinders user-friendly modelling, which is not regarded in respective publications.

Another frequently used modelling approach is **Hierarchical Task Decomposition**. Consider the example marked blue in Figure 2.1d: A sealed bottle with a bar code label can be assembled by mating the labelled bottle with a lid, if the bar code has been affixed in advance. The blue edges form a **Hierarchical Task Network**, where assemblies or tasks (blocks) are split up recursively until parts or basic actions are reached (circles). An extension towards formal operators to enforce sequential execution and to express parallelism has been used for task sharing by Roncone et al. [103]. These models can be

	Parallelism	Complexity	User-friendly Modelling	
Operation Sequence	✗	low	LfD	[67][64]
Precedence Relations	✓	medium	(✗)	[81][86][78][90][97][16]
Temporal Constraint Networks	✓	high	✗	[73][40][126][112]
Hierarchical Decompositions	✓	medium	LfD (✗)	[103] [30][58][47][39]
Finite State Machines	✗	high	LfD (✗)	[44] [60]
Probabilistic Models	✗	very high	(LfD) ✗	[27][65][88][89] [9][114][23][71]
Petri Nets	✓	high	(✗)	[25]

Table 2.1.: Properties of task models used for human-robot teaming (✗ = aspect not considered, (✗) = aspect partly considered, ✓ = aspect considered, LfD = Learning from Demonstration)

learned from demonstrations [49]. **AND/OR Trees** are a common model with similar expressiveness from the assembly planning domain [53]. A *hyperarc* combines two sub-assemblies or parts (AND). Each node that is not a leaf of the tree may have several outgoing hyperarcs – for planning task execution, exactly one outgoing arc must be chosen in each node while descending into the tree from its root (OR). AND/OR Trees are the basis of several human-robot teaming methods [30, 58, 47, 39]. These approaches do not consider how end-users can supply the system with new task descriptions. However, AND/OR Trees can be generated automatically similar to precedence graphs, e.g. using the ‘assembly by disassembly’-paradigm [118].

Hamabe et al. have shown, how **Finite State Machines** for human-robot task sharing can be learned from human demonstrations [44]. States can encode the static status of the assembly as well as dynamic human motions in their modelling scheme. State transitions are triggered based on perception results and cause robot actions as an output. This complex representation can cover different ways to accomplish a task. Like a set of feasible operation sequences, it does not express explicitly to what extent operations may be executed in parallel. The state machine for collaborative screw assembly used by Jülg et al. [60] combines robot perception and action similarly. In contrast to the approach of Hamabe et al., states do not describe detailed events related to the state of workpieces. They encode more general actions or sub-programs as e.g. checking the human pose, selecting a screw to approach, executing the trajectory to this screw or fastening it. The approach relies on trajectories that are taught by hand-guiding the robot, but a user-friendly way to model such rather specific state machines is not addressed.

Several **Probabilistic Models** are based on the basic structure of state machines but extend them towards handling uncertainty. Uncertainty can e.g. emerge from noisy sensor data or incomplete knowledge on human intentions. Lenz et al. have shown how composite **Hidden Markov Models** (HMMs) can be used to analyse human progress within a workflow as a basis for planning of supportive robot actions [71]. To this end, HMMs for several basic actions are trained with corresponding hand motion sequences and arranged to form a complex task manually. **Dynamic Bayesian Networks** (DBNs) are a generalisation of HMMs. In the work of Baraglia et al. [9], a two time-slices DBN encodes task knowledge in the probability of taking an action in a certain state and in the distribution over the next state given an initial state and action. The issue of supporting the user in specifying these probabilities is not addressed here. Other approaches rely on variants of **Markov Decision Processes** (MDPs) for robot action selection [27, 65, 114, 88, 23, 89]. A MDP is generally defined by a tuple $\{S, A, T, R\}$. In the aforementioned publications, task progress in terms of the physical world state $s \in S$ is a fully observable variable. The robot is capable of a finite set of actions A . Ways to achieve a task are encoded in the reward function $R : S \times A \rightarrow \mathbb{R}$ by assigning high rewards to the feasible or particularly favourable actions in each state. Given the current state s and a robot decision on its next action a , the next state s' depends on what the human does while the robot performs a – this variance in human actions is captured by the state transition function T , which gives a probability distribution over s' for each combination of s and a . Worker preferences can be taken into account by formulating human goals and strategies as an additional, unobservable variable in mixed-observability MDPs [88] or partial-observability MDPs (POMDPs) [23]. Similarly, an estimate on human trust in success of certain robot operations can be integrated [27]. Their stochastic nature makes MDPs highly complex and creating appropriate models for practical applications manually can be challenging and tedious [132, 88]. Koppula et al. have proposed a learning approach for MDPs based on object affordances [65]. The methods of Nikolaidis et al. [88, 89] and Chen et al. [27] support learning for parts of the model, e.g. by adjusting rewards and transition probabilities to match the habits of individual workers using cross training [89] or detecting their strategies in observations of human co-working [88]. The remaining approaches rely on manual specification [114, 23]. Similar to state machines, the above task models describe processes in a sequential manner and do thus not encode parallelism explicitly.

Casalino et al. have defined basic Time **Petri Nets**, e.g. for human actions, robot actions, collaborative actions or part transports by mobile robots [25]. Aside from resource allocation, these component Petri Nets model the act of waiting for a partner explicitly, resulting in a high overall task model complexity. The components can be assembled into more complex co-working processes using sequential, parallel or alternative connections. Pictograms for each of the component Petri Nets target user-friendly modelling in theory, but Casalino et al. do not focus on this aspect explicitly.

The aforementioned approach of Casalino et al. relies on manually specified, complex component Petri Nets that the robot system needs for decision-making [25]. Once these components are defined, task modelling is performed by composing processes of several

actions using a simplified descriptive formalism – a larger Petri Net for the overall task is then deduced by mapping from pictograms to the component Petri Nets they represent. Similar **Hybrid Task Models** with an automated mapping between some high-level, human-legible task representation and a robot-specific model have also been proposed by other authors. Hawkins et al. have shown how to transform an input AND/OR Tree into a Bayes Net to track human actions within a task with variants [47]. Similarly, the approach of Roncone et al. embeds the root nodes of a hierarchical task network into partially observable MDPs to negotiate role assignment [103].

2.2. Decision-Making for Task Allocation

With the task model given, the second important pillar of dynamic task sharing is a strategy that enables meaningful robot decisions on task allocation. Several criteria to base decision-making algorithms on have been proposed. This section identifies main categories among these approaches and classifies related publications accordingly. The classification result is summarised by Table 2.2. In this table, works are moreover divided into approaches to static and dynamic task allocation. Here, *static* means that the ordering of actions and their assignment to agents are completely determined in advance of task execution. By contrast, *dynamic* task allocation supports repeated, situational decisions during execution as motivated in Chapter 1. The scope is limited to publications that consider human and robot as peers with equal importance to the task. *Semi-dynamic* methods allowing agents to make decisions only within a fixed role are therefore not taken into account – approaches of this category limit either human [61] or robot [48, 71, 47] to the subordinate assistant role of handing parts at the right time.

A large group of publications puts emphasis on technical aspects of the process or robot system. In this category, several methods rely on the use of **capability indicators** for task allocation decisions. *Capability indicators* rate to what extent some operation τ is suitable for execution by either human or robot. To this end, calculation of such indicators typically condenses resource suitability according to several criteria into a real-valued score $c_H(\tau)$ for a human and $c_R(\tau)$ for the robot – then, the higher of both scores indicates that τ should be assigned to the corresponding agent. An early concept of Beumelburg [16] defines a comprehensive criteria catalogue and indicator computation procedure for cooperative assembly planning which later works build upon [18, 100, 109]. Respective criteria may e.g. cover properties of objects involved in operations: Parts that are heavy or easy to grasp suit the strengths of robots, whereas compliant, sensitive or fragile objects require human dexterity. Concrete indicator values emerge from a manual expert rating process during which a characteristic (e.g. ‘low’, ‘medium’ or ‘high’) with an associated real valued indicator per agent is chosen for each pair of a category and operation [16]. By contrast, indicators can also be determined automatically, e.g. by matching the part weight and dimensions against the maximum possible payload and gripper opening of the agent in question [78, 81]. Aside from these technical aspects, risks for human health, especially regarding ergonomic strain, are another major factor. This aspect can be quantified by applying the Strain Index (SI) [97] or Rapid Entire

	static	dynamic
Technical focus – e.g. makespans, ergonomics, properties of parts and hardware		
Capability indicators	[86][78][81][18][75][84][24] [8][97][100][58][109][26][16]	[90]
Workspace observation	—	[60][30][9][44]
TCN scheduling	—	[73][112]
Social/Cognitive focus – e.g. preferences, habits, trust, knowledge, communication		
TCN scheduling	[40]	[126]
MDP policy-following	—	[27][103][88][89] [23][65][114]
Symbolic planning	[85][33]	[111]

Table 2.2.: Different decision-making strategies for static/dynamic task allocation in literature

Body Assessment (REBA) [78, 24] techniques to human motions that constitute an operation. With the final assembly sequence given, rules based on capability indicators can be used for task allocation [78, 100]. In the case of more complex task models (e.g. the precedence graph or AND/OR Tree of a production process), a schedule that maximizes the use of individual capabilities while minimizing e.g. cycle times or energy consumption can be calculated by multi-objective optimisation. Solving the optimisation problem may require estimates for the duration of operations when carried out by either agent – estimates for robots can be obtained from simulations, whereas motion time systems as Methods Time Measurement (MTM) provide standard times for workers to complete some action [75, 8, 109]. The final schedule can then be computed using variants of tree search algorithms [24, 58], genetic algorithms [86, 8, 26, 16] or mixed-integer linear programming [97]. Optimisation approaches may even include planning of the spatial cell layout in addition to task allocation [84]. All the aforementioned planning methods perform static task allocation – however, a dynamic mode of task sharing can be achieved by solving the optimisation problem repeatedly depending on recent events [90].

Workspace observation, i.e. monitoring of events, parts and humans, is the foundation of decision-making for a second class of approaches with a technical focus. The state machines of Hamabe et al. output the next robot operation when transitioning into the next state [44]. Transitions depend on recent observations of human actions or the workpiece state. The robot is thus able to take part in the task dynamically within the bounds of possible assembly sequences that were demonstrated during a learning phase. Darvish et al. have shown how to ground skill input parameters by simulating operation outcomes depending on the current situation in the workspace [30]. Then, operations can be parametrized and allocated online to the agent yielding minimum execution times. Given consecutive observations of parts in the workspace, Baraglia et al. calculate a so-called *object detection likelihood* [9]. This measure judges the stability

of object recognition results – it increases as long as some object is sensed and decreases during time steps when it cannot be recognized. The approach seeks to assign parts a low likelihood when interaction of humans with objects renders perception noisy. The robot can then choose the operation that will most likely succeed from the set of all feasible operations according to high detection likelihood values. Jülg et al. have proposed a different way to reduce interference of human and robot actions [60]: In their work, trajectory execution is aborted, when online collision checks based on RGB-D camera images predict a collision with the human. The robot can then switch to an operation with a trajectory that is collision-free according to the current workspace situation.

Temporal constraints that need to be satisfied during task execution are another decision criterion for task allocation. Such constraints can be specified as a part of TCNs (Section 2.1). They may e.g. limit the amount of time that the overall task execution should take while also bounding the duration of individual operations. Approaches that aim to dispatch variations of TCNs by satisfying all constraints typically make decisions based on a special encoding of the problem [112]: The input TCN is compiled into a representation that aggregates all feasible schedules for all task allocation possibilities offline. This encoding can then be used for situational online decisions that take the consequences on overall task feasibility into account [73, 126, 112]. These approaches are thus explicitly tailored to enable dynamic, on-the-fly decisions through TCN scheduling.

Preserving consistency with temporal constraints implies reflecting other team members’ decisions implicitly. Some TCN-based approaches furthermore incorporate explicit agent preferences [40, 126]. This brings us to the next major category of publications that put emphasis on social and cognitive aspects. Aside from the TCN scheduling approaches with preferences that follow an optimal schedule calculated offline [40] or anticipate workers’ habits dynamically by re-scheduling [126], **following MDP policies** is a widely accepted technique. This method is, of course, strongly intertwined with using MDPs for task modelling. In general, the robot system can derive a *policy* π that assigns an action to every possible world state for a given MDP. Policies are designed to output optimal actions in each situation covered by the state set S in a way that maximises the sum of future expected rewards [89]. This fast lookup renders policy following suitable for dynamic task allocation. Depending on the concrete formulation of the MDP, particularly in terms of the reward function R , different social and cognitive aspects can be covered (cf. also Section 2.1). For instance, rewards or hidden state variables can be defined to reflect *human preferences and strategies* [88, 23, 89]. Similarly, the method of Koppula et al. adjusts the robot policy depending on whether humans prefer sticking to their past habits or adapt to prior robot actions [65]. Other authors have shown how to integrate task execution with transparent, *socially acceptable communication* [103, 114]. The system of Chen et al. [27] is furthermore able to build *trust* by favouring actions that the human partner believes the robot will be capable of.

The final class of approaches with a focus on aspects of human cognition uses **symbolic planning** to allocate the operations of a task to a human or robot. These approaches trace back to the *Hierarchical Agent-Based Task Planner* (HATP) [32, 3], a HTN planner with special features for human-robot interaction (e.g. support of social rules throughout

the planning process [68]). A recent method based on the HATP considers human *prior task knowledge* in the planning process but negotiates the whole plan statically before the execution process [85]. Although the system of Devin and Alami [33] is able to supply a human partner with information online depending on an estimate of human *situational knowledge* (e.g. after a period of absence), this method still relies mainly on plans that are completely communicated beforehand. Sebastiani et al. have later shown, how multiple HATP-generated, socially acceptable plans can be merged into a conditional Petri Net Plan that enables dynamic negotiation of task allocation decisions [111].

2.3. Coordination Mechanisms

The last perspective that we will take on related publications surveys the coordination mechanisms they employ. The scope is limited to approaches that support dynamic task allocation according to Table 2.2. Against the background of cost-efficient SME applications, the extent of sensory input is a key factor. Partial observability of the physical world state induced by a limited sensor set is thus especially relevant (Section 1.3). Therefore, the following analysis does not consider methods in-depth that presume full world state observability as a strict formal requirement. This applies particularly to systems with MDPs as the underlying model [65, 114, 89]. Similarly, the partial- and mixed-observability MDPs of Chen et al. [27] and Nikolaidis et al. [88] enable reasoning on the unobservable human mental state, but still demand full world state observability explicitly. We will see hereinafter that concrete coordination mechanisms greatly influence the feasibility of transitions between different modes of flexible teams (cf. Section 1.2). Table 2.3 compiles information on the implementation of coexistence, cooperation and collaboration in addition to support of partial observability in the remaining works to the best of the author’s knowledge. Besides the below in-detail explanations, the classification is based on following criteria: A system enables cooperation, if and only if human and robot can work in parallel. A workflow during which agents take turns by handling operations one after another (also referred to as *sequential cooperation* [12]), is classified as partially considering the cooperation mode. Collaboration is assumed fully supported if the system can coordinate task models that include joint operations for several agents automatically. If different roles within a collaborative sub-task need to be modelled explicitly as on operation each, collaboration is only partly addressed.

A first class of approaches is characterized by **explicit negotiation** of upcoming task allocation decisions. The conditional plans of Sebastiani et al. [111] support cooperation, i.e. parallel execution of actions by human and robot, based on the underlying symbolic planner [3]. Collaboration in terms of joint actions of several agents is not considered. Explicit negotiation introduces a tight robot-to-human dependency – this hampers productive coexistences as the robot system loses its capability of acting as soon as the human leaves and stops answering to requests. Although further sensory input is not regarded explicitly, the approach can generally reduce the amount of sensing to a limited number of conditions at crucial decision points in the plan. Partial observability is thus theoretically manageable to some extent. Similarly, the approach of Roncone et al. [103]

	Coexistence	Cooperation	Collaboration	Partial Observability
Baraglia et al. [9]	●	●	○	●
Darvish et al. [30]	○	◐	●	○
Gopalan et al. [23]	○	◐	○	◐
Hamabe et al. [44]	○	◐	●	○
Jülg et al. [60]	●	●	○	○
Levine et al. [73]	◐	●	◐	○
Nikolakis et al. [90]	●	◐	●	○
Roncone et al. [103]	●	◐	◐	○
Sebastiani et al. [111]	○	●	○	◐
Shah et al. [112]	◐	●	◐	○
Wilcox et al. [126]	◐	●	◐	○

Table 2.3.: Properties of dynamic task allocation approaches regarding modes of flexible teaming and handling of partial observability (○ = not considered, ◐ = partly considered, ● = considered)

manages coordination solely through mutual communication, i.e. partial observability of the world state is not made a subject of discussion. The method is based on asking the human to commit to operations via ‘yes’/‘no’-questions that can be answered verbally or using a graphical user interface. In case of human commitment, the robot will eventually ask to confirm completion of the operation. Coexistence is assumed feasible as missing answers are explicitly mentioned to be considered in the planning process. However, human and robot cannot work in parallel as the POMDP planner results in sequential, turn-taking alike execution of operations by either human or robot. Collaboration is partly feasible but needs to be modelled explicitly as distinguished operations for both agents in the task model.

Variants of **turn-taking** or **sequential cooperation** in the wider sense have also been realised by other authors without explicit negotiation. All the following approaches match the above definition for partial consideration of the cooperation mode as taking turns merely enables sequential work rather than parallel execution. Gopalan et al. [23] rely on strictly alternating turns of human and robot, resulting in structured sequences of actions and observations for planning with POMDPs. A robot action must be followed by an observation produced by the human. The robot system can then select its next step based on the underlying optimal policy. Such workflows can neither enable coexistence, nor collaboration. Partial observability is considered partly with regard to the human mental state and object properties that are generally unobservable with robot sensors – however, these aspects do not cover effects induced by moveable eye-in-hand cameras, where objects are (un-)observable depending on the current sensor pose in the workspace. Darvish et al. [30] focus on more flexible workflows with a less rigid structure. The proposed architecture informs humans about optimal task allocation using a display. Re-planning is triggered by observations of world state changes and human actions indicating

a deviation from the optimal plan. Robot planning supports collaborative actions but does not consider situations in which the human is unavailable. Coexistence is thus not supported, and cooperation is limited by the fact that the robot stops as soon as a human action is sensed. Similarly, the schedules executed by the system of Nikolakis et al. [90] do not incorporate parallel execution of operations. Their approach does however support coexistence by integrating agent availability into the task allocation process, and by considering unavailability of resources as an additional reason for rescheduling. While perception and communication are considered black boxes in the framework of Nikolakis et al. [90], the method of Darvish et al. [30] relies on rich sensor data for world state and action monitoring without addressing the issue of partial observability. This also holds for the system of Hamabe et al. [44], where events observed with several RGB-D cameras trigger robot actions that mainly focus worker support in collaborative operations.

Mutual information, e.g. in terms of robots sharing their ideas for optimal task allocation [90, 30] or humans confirming successful execution of operations [103], are central elements in most of the aforementioned approaches. The following group of methods puts even stronger emphasis on fine-grained, **explicit information** about the start or end of an agent’s operations: Agents make decisions freely and inform their partners reliably, thus stating commitment to operations rather than negotiating or taking turns. To this end, software buttons [103] or text-based messages [40] via graphical user interfaces, hardware buttons [60] and verbal communication [112] have been used to provide robot systems with complete knowledge on humans’ choices and task advancement. The approach of Jülg et al. [60] fully enables coexistence and cooperation based on human confirmation of completed operations. It does, however, not provide means for meaningful, time-efficient collaborative operations. By contrast, TCN-based scheduling methods as e.g. that of Shah et al. [112] can reduce agent idle times and thus enable execution of collaborative acts when explicitly specified in the task model (e.g. object handovers without either agent having to wait unnecessarily). Respective planners require precise information about human commitment and timing by design. They do therefore not consider partial observability. Based on the information about human commitment, TCN-based schedulers can handle coexistence partly – yet the loss of time induced by absent agents may render solving the underlying constraint satisfaction problem infeasible within the prescribed maximum completion time.

Commitment to execute an operation can certainly not only be established by communicating explicitly, but also by monitoring one’s partner’s actions. This mode of **implicit information** is used by the temporal plan dispatchers of Wilcox et al. [126] and Levine et al. [73] which realise workflows similar to the method of Shah et al. [112]. The shortcoming of these TCN-based approaches with regard to coexistence has been addressed in the context of Levine’s method [73] by negotiating a relaxation of temporal constraints with the human, when necessary [62]. Still, full coexistence stays infeasible – the issue of robot actions during a phase in which the human does not react to negotiation attempts is not discussed. Compared to strongly structured workflows that are shaped by mandatory negotiation, turn-taking or frequent explicit information, implicit information gathering is a less constraining, yet powerful mechanism. Deduction of task progress from

observation of the world state or human actions is thus a unifying component across most task sharing approaches: E.g., negotiation based on symbolic planning still needs stable means to detect operation effects and turn-taking similarly requires a way to observe when turns end. Robust perception is even more relevant to MDP-based methods that formally rely on full observability [65, 114, 89]. Obtaining such information therefore usually involves comprehensive sensor systems in recent prototype implementations, e.g. several calibrated RGB-D cameras [60, 44], wearable smart devices [30] or specialized motion tracking systems [88] for detecting human actions. In this respect, the prototype of Baraglia et al. [9] differs significantly. It supports a proactive mode in which the robot handles operations autonomously as soon as they become feasible. This autonomy yields full support of coexistence and cooperation but does not enable collaboration on joint actions. Feasibility is here tied to the stability of perception results for parts related to the operation (cf. Section 2.2). If objects are not visible (e.g. due to occlusions caused by human motion in the workspace), corresponding operations will not be considered for execution temporarily until stable rediscovery. Addressing the issue of parts getting out of sight and reappearing in the robot world model later matches the situation of moving eye-in-hand cameras with a limited field of view. Partial observability in the sense of the definition in Section 1.2 is hence manageable with this approach.

2.4. Conclusions

The key insights gained from related work can be summarised as follows: Several task models have been used in the context of human-robot task sharing (Section 2.1). They mainly differ in their complexity. Task representations with low to medium complexity as e.g. operation sequences, precedence graphs or hierarchical task decompositions are hypothesized to be sufficiently comprehensible for investigating end-user modelling. They are moreover strongly related to the target industrial applications as their origins lie in the assembly planning domain [54]. In this class, precedence graphs stand out as the most frequently used model that furthermore encodes parallelism explicitly. A drawback of these representations is that they merely encode task structure, i.e. they can be seen as a task mental model only. This limitation in expressiveness is overcome by more complex representations: Finite State Machines can e.g. encode preferred workflows learned from observation [44]. Variants of probabilistic Markov Decision Processes have been shown to even enable reasoning on the human mental state – this way, teamwork mental models in terms of human preferences [88], communication [114] or trust in the robot [27] are interwoven with a description of the task structure. However, these powerful models can hardly be created manually [132, 88], thus rendering them impractical with regard to the goal of graphical programming by domain experts. Few approaches use hybrid task models to strike a balance between comprehensibility and expressiveness [25, 103]. These works involve two modelling layers, where the task mental model can be dealt with by end-users, whereas creating team mental models is left to expert designers.

Given a task model, task allocation decisions can be made in different ways (Section 2.2). Two main categories can be distinguished (Table 2.2): A first group of approaches focusses

technical aspects to create efficient processes with optimised makespans by maximizing the use of agents' individual capabilities. By contrast, adaption to human cognitive processes, e.g. in terms of habits, trust or knowledge, is the main goal of the second category. Technical aspects are predominantly encoded by capability indicators. These numerical indicators quantify suitability of humans and robots for operations. They cover e.g. human motion ergonomics [78, 97], feasibility for robots with respect to their limited dexterous skills [16] or execution time for one or the other agents [30]. Similar to this thesis, these metrics often target the needs of partial automation in SMEs. Several methods based on capability indicators rely on comprehensible task models (e.g. [78, 97, 16]). But these approaches mostly foster static offline task allocation, as optimisation can be time-consuming and hence infeasible for dynamic decisions (cf. the planning times for realistic use cases reported by Pearce et al. [97]). By contrast, dynamic approaches with a technical focus are rather based on more complex models as state machines [60, 44] or formulations of temporal constraint satisfaction problems [73, 112]. Similarly, dynamic approaches in the second category are dominated by complex MDPs. As creation of MDPs often involves learning from human demonstrations, this decision-making strategy is hardly feasibly without time-consuming training procedures.

Depending on the coordination mechanism (Section 2.3), workflows emerging from recent dynamic task sharing approaches are mainly characterized by bidirectional negotiation [111, 103], turn-taking [23, 44], explicit [60, 112] or implicit [73, 126] unidirectional information about agents' progress. Implicit information in terms of action and workspace observation can also serve as a coordination mechanism [9]. Of course, robust and reliable perception plays a vital role for most systems, particularly for those formally relying on full world state observability [65, 114, 89]. Recent prototypes are therefore mainly based on extensive sensor systems incorporating multiple cameras [60, 44], smart wearable devices [30] or even complex motion tracking systems [88]. If considered at all, strategies to manage partial observability are usually directed towards reasoning about the unobservable human mental state rather than partial workspace observability. An analysis of workflows and coordination mechanisms used with dynamic task allocation schemes shows that each approach has at least one of the following properties (Table 2.3): (i) Tight coupling, e.g. through mandatory negotiation or temporal constraints, limits practicability of coexistence, (ii) cooperation happens in a sequential, turn-taking like manner without making use of parallel execution (iii) or coordination of collaborative steps is not considered, unless represented as distinct, complementary operations in the task model – however, this means increased modelling efforts, as splitting operations into parts manageable by either partner requires expert knowledge on robot capabilities.

Based on these lessons learned, a need for research regarding the application-oriented goals outlined in Chapter 1 can further be substantiated from the scientific point of view: To the best of the author's knowledge, there is no approach to dynamic human-robot task sharing in literature that enables fully flexible teaming. Dynamic task allocation has mostly been achieved by relying on complex, hardly human-legible task models and on full world state observability as achieved with extensive sensor systems. There is thus a gap in knowledge about application-oriented flexible teaming systems that enable

graphical programming while also being robust to a limited amount of sensory input. All in all, this thesis seeks to contribute to prior knowledge by describing a human-robot teaming framework (i) that realises the full spectrum of flexible teaming across the modes of coexistence, cooperation and collaboration, (ii) that is based on a sufficiently comprehensible task model for graphical programming while still enabling dynamic task allocation (iii) and that can handle partial observability of the world state to reduce the amount of required sensors.

Task Modelling for Human-Robot Teams

3.1. Skills for Human-Robot Teams	31
3.1.1. Domain Definition	32
3.1.2. Skill Graph Structure	35
3.1.3. Benchmark Domain	38
3.2. Shared Task Model Generation	43
3.2.1. Graphical Modelling of Precedence Graphs	43
3.2.2. Annotation with Operation Pre- and Postconditions	46
3.3. Task Execution Principle	47
3.4. Conclusions	48

THE following chapter addresses the issue of establishing a shared task mental model among human and robot. Task mental models encode procedural knowledge on a task [82], and they are a prerequisite for efficient teamwork [74]. Precedence graphs are a suitable task representation for evaluating flexible teaming as they encode parallelism and can be claimed sufficiently comprehensible for end-user modelling (Section 2.4). They will therefore be used as an approximation of task mental models from here on. Having the goals and research questions of Chapter 1 in mind, the approach described hereinafter takes inspiration from literature on graphical, task-level robot programming in general, and on architectures for robust and reusable robot skills in particular.

Hiding low-level complexities of robot programming by introducing levels of abstraction is a common way to reduce the expertise required to instruct robot systems [117, 98, 4, 120]. This idea is reflected by the three-layered structure of the proposed task modelling framework (Figure 3.1). The user is supplied with a set of symbolic skills on the highest level of abstraction (Section 3.1). Skills represent complex operations that form a union within the application domain from the human point of view, as e.g. picking and placing an object, mating two parts or applying a tool to some workpiece. They accept mostly object-related parameters that humans with knowledge on the domain can specify, e.g. the affected parts or a part goal position within a parts bin. After parametrisation, skills are ordered to form a precedence graph that describes a task using a graphical user

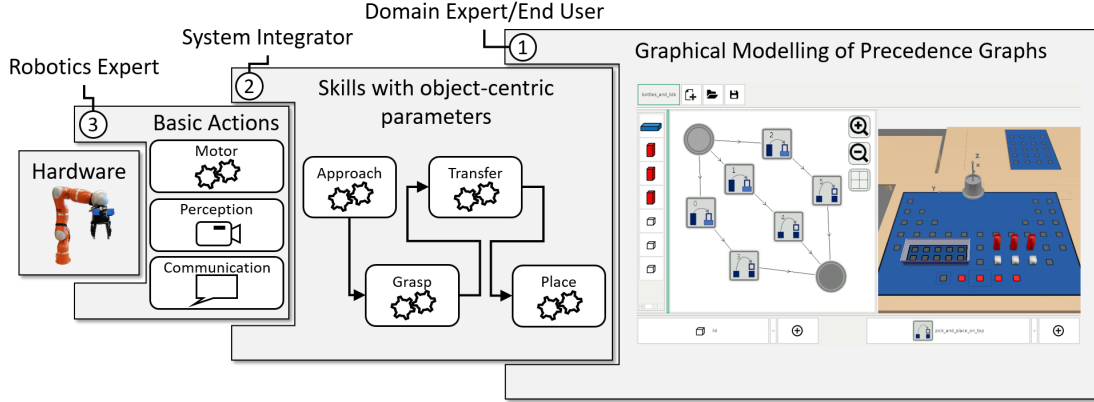


Figure 3.1.: The proposed task modelling framework is three-layered: Level 1 enables process-oriented graphical end-user modelling of precedence graphs using abstract skills. On Level 2, the domain-dependent skills are defined as graphs of basic actions as specified by a system integrator. The gap to robot execution of skills is bridged by robotics experts on Level 3 where hardware implementations for symbolic basic actions are provided.

interface (Section 3.2). It is important to notice that modelling on Level 1 is intended to provide a purely process-oriented view onto tasks. There are no assumptions on the feasibility of parametrised skills for either agent, as this would e.g. require expert knowledge on the robot operating range as defined by kinematic properties.

On the second level of abstraction, individual skills are defined as graphs of so-called basic actions. Skill graphs must have a specific structure that is tailored to suit the needs of human-robot coordination later on. In particular, skills must be robust to execution failure as far as possible and report their outcomes to the planning component reliably. Moreover, the set of basic actions should support communication as well as control of actuators and sensors – therefore, specification of basic actions and modelling of skills for a domain is supported by a system integrator who is familiar with these requirements. On the lowest Level 3, hardware-independent basic actions are mapped to concrete hardware commands by a skill execution engine. To this end, an implementation for each basic action is provided by a robotics expert for the target hardware setup. Skill graphs thus bridge the gap between an abstract, high-level process description and robot control – they model the role of a robot within a potentially collaborative operation.

The above considerations put emphasis on hardware independence, convenient extension of the skill set, and on end-user modelling of tasks – these are beneficial properties from the point of view of a human operator who needs to share knowledge with the system. By contrast, there are different demands regarding a task model when taking the perspective of robot system design. The precedence graph of symbolic skills is especially not sufficient for a robot to observe task progress. Preconditions and postconditions are commonly used in symbolic planning as well as skill frameworks (e.g. [98, 120]) to render the prerequisites and effects of skills observable – the proposed skill formalism adopts this strategy by supporting automatic generation of conditions per skill (Section 3.2.2).

Similar layered skill frameworks (e.g. [98, 5, 120]) as well as graphical tools for industrial task-level programming (e.g. [117, 96]) have recently been proposed. The below approach differs from prior art by (i) considering complex tasks encoded by precedence graphs rather than operation sequences on the highest level of abstraction (ii) and by focussing on process descriptions rather than expecting the user to model task control flow including robot perception manually. Recent works on dynamic teaming take a similar view on task definition, but do not consider the benefits of graphical programming and structured transitions from skills to commands for different hardware platforms (cf. Section 2.1). Against this background, the core goal of this chapter is to provide a formal framework that is particularly tailored to suit the needs of human-robot teaming. Skills can be composed of reusable, hardware-independent actions. New skills designed in line with this structure will integrate instantaneously with a widely generic graphical programming tool – as soon as they are added, required parameters can be queried from users via auto-generated forms. A mechanism for automated prediction of skill outcomes enables visualization of skill effects as well as integration with the coordination algorithm for human-robot teaming (Chapter 5) by means of pre- and postconditions. Section 3.1.3 shows, how this structure can be applied to a concrete domain for benchmarking human-robot shared tasks. Features and limitations are finally discussed in Section 3.4.

3.1. Skills for Human-Robot Teams

The framework described hereinafter takes inspiration from the graph-based skill definition of Andersen et al. [5]. Their work has introduced a formalism to represent skills as hierarchical graphs of symbolic basic actions. Graph edges establish a control and data flow throughout the skill – starting from user-specified input parameters, a generic skill execution engine can then follow the control flow and map the encountered basic actions to their implementations for concrete hardware setups. Building upon this formalism, the list of skill graph nodes will be denoted by

$$s^B = (b_{s,1}, b_{s,2}, \dots, b_{s,|s^B|}) \quad (3.1)$$

in the following. Each graph node instantiates one out of a finite set of parametric *basic action templates*. These templates model small, non-interruptible process steps within the application domain (Section 3.1.1). While partly cutting down on the generality supported by the skills of Andersen et al. [5], their formalisation is extended towards a specially shaped control flow s^{flow} and data flow s^{data} in Section 3.1.2 (Figure 3.2). The resulting graph structure renders skills particularly suitable for human-robot teaming and graphical task modelling: In the ideal case, execution follows a path of successful basic actions. It is guaranteed by definition of the framework that input objects can reliably be transformed into their expected state after successful execution of a basic action. Based on this prediction, postconditions can automatically be constructed from input parameters by concatenating successive basic action effects along this path. Compared to manual definition of conditions on the skill level rather than for basic actions (e.g. [98]), this shifts effort to the expert level of basic action implementation, but therefore opens

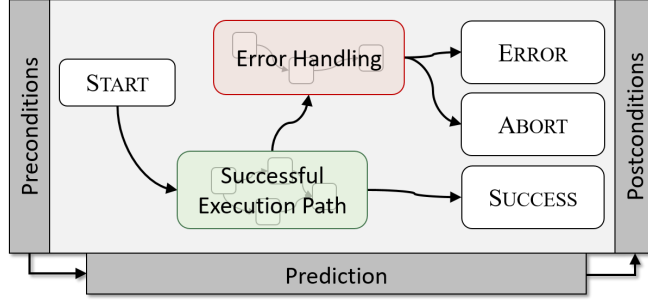


Figure 3.2.: Skills for teaming are specified as graphs of basic actions with a special structure: Control flow along the successful execution path (green) enables prediction of pre- and postconditions. Error handling (red) either leads to an ABORT after reverse execution, or ultimately indicates failure by terminating in the ERROR state.

skill modelling to persons with less detailed knowledge. If execution of a basic action fails, skills branch out into error handling procedures. Following the argumentation of Laursen et al. [70], two major situations are distinguished: For cases in which intermittent changes are reversible, skills contain a sequence of actions to undo prior steps. Having restored the state before execution, the ABORT final state leaves a superordinate coordination layer the decision to retry or re-plan. If reverse execution fails or is generally infeasible (e.g. after erroneously breaking or bending a part out of shape), the ERROR state indicates severe failure. This may e.g. result in a request for human help.

3.1.1. Domain Definition

An application domain for human-robot teams is mainly characterized by the objects to be manipulated and by the basic process steps that may occur. It is therefore reasonable to start skill definition with a formal specification of these two components:

Objects are described by a set of *aspects* $A = \{a_1, a_2, \dots, a_{|A|}\}$. Each aspect covers an object property that is relevant to the domain (e.g. the type, weight or position of a part in the workspace). Aspects can take values from their associated *aspect domains* D_i ($i \in 1, \dots, |A|$). The object position may e.g. be taken from $D_{\text{pos}} = \mathbb{R}^3$. Let $\bar{D} = D_1 \times D_2 \times \dots \times D_{|A|}$ denote the cartesian product of all aspect domains. Then, an *entity* of the physical world is fully specified by an *object state* $e \in \bar{D}$ that consists of a value for each aspect. For example, parts are defined by their position and type in some example pick-and-place domain $\bar{D}_{\text{pap}} = \mathbb{R}^3 \times \{\text{bottle}, \text{lid}, \text{tray}, \dots\}$. We will see later-on that it is often necessary to compare task object states. This is achieved by *aspect comparators*, denoted by \approx_{D_i} for each $i \in 1, \dots, |A|$. Formally, \approx_{D_i} is a relation that encodes similarity of values from D_i . A criterion to determine whether $d \approx_{D_i} d'$ ($d, d' \in D_i$) is assumed to be provided by the robotics expert during the implementation of Level 3. For example, one can say that $x \approx_{D_{\text{pos}}} y$ if and only if $\|x - y\| < \epsilon_{\text{pos}}$ for some application-dependent precision ϵ_{pos} when comparing object positions x, y from $D_{\text{pos}} = \mathbb{R}^3$. Aspect comparators can be assembled into an *object state comparator* $\approx_{\bar{D}}$ for elements of \bar{D} , with

$$(d_1, d_2, \dots, d_{|A|}) \approx_{\bar{D}} (d'_1, d'_2, \dots, d'_{|A|}) \Leftrightarrow \forall i \in \{1, \dots, |A|\} : d_i \approx_{D_i} d'_i. \quad (3.2)$$

Basic Actions are symbolic denominators for building blocks of more complex skills. In contrast to the common understanding of skill primitives as pieces of robot motion

according to a certain control strategy [80] (e.g. for establishing contact states [46]), basic actions are here more generally defined as the smallest meaningful and non-interruptible action units within the domain. They may e.g. realise the well-known approach, detach and transfer motions for a pick-and-place domain. The notion of basic actions is however not limited to primitive movements – a transfer action may e.g. encapsulate a motion planner, and a grasp action implementation might rely on a grasp planner to find a suitable pose for picking up an object. Depending on the concrete domain, a set of building blocks may possibly also include more complex handling actions as fastening a screw or mating of distinguished parts. Implementations of respective actions can then e.g. serve as a transformation layer that bridges the gap between task-level parameters and parametrisation of skill primitive networks [119] for force-based manipulation. To comply with the special needs of our human-robot teaming scenario, the skill framework used in this thesis distinguishes the following types of basic action semantics:

- *Motor Actions* B^{motor} control actuators of any kind used by the hardware setup, e.g. the robot arm and a gripper.
- *Sensor Actions* B^{sense} trigger sensors, in our case e.g. the eye-in-hand RGB-D camera attached to the robot hand.
- *Communication Actions* B^{comm} support explicit communication, e.g. waiting for the confirmation that an action that is unobservable with the camera system has been done by a human partner.

Any implementation of basic actions for a concrete hardware platform is assumed to indicate either success or failure after being called by the skill execution engine.

The formal definition of skills as well as basic actions $b \in B = B^{\text{motor}} \cup B^{\text{sense}} \cup B^{\text{comm}}$ relies on *parameters* that are needed to control the behaviour of b , e.g. a concrete position to approach or the goal robot pose of a transfer. A parameter p with

$$p = (p^{\text{name}}, p^{\text{type}}) \quad (3.3)$$

consists of a name p^{name} and the type p^{type} . The name is a string used for identification within sets and visualization during graphical task modelling. Possible types are taken from a set PD of potential domains of parameter values. The set of object states \bar{D} is a mandatory element of PD as parametrisation with objects is a key goal of task-level programming. Continuing the pick-and-place example, parameter types may e.g. be given by $PD_{\text{pap}} = \{\bar{D}, \mathbb{R}^3\}$. We can formulate parameters as (`moved_object`, \bar{D}_{pap}) or (`goal_position`, \mathbb{R}^3) for basic actions of a transfer skill with this set.

Based on this concept of parameters, each basic action template b is given by a 3-tuple of input parameters b^{in} , output parameters b^{out} and a prediction function b^π , i.e.

$$b = (b^{\text{in}}, b^{\text{out}}, b^\pi). \quad (3.4)$$

Each skill graph node with index $i \in \{1, 2, \dots, |s^B|\}$ (Equation 3.1) instantiates one of the parametric basic action templates, i.e. $b_{s,i} = b \in B$. In other words, templates are copied

into the node list, so that each $b_{s,i}$ has components $b_{s,i}^{\text{in}}$, $b_{s,i}^{\text{out}}$ and $b_{s,i}^\pi$ as defined for basic actions. We can now introduce *values* of parameters, which determine the individual behaviour of each action template instance at execution time: For an instance $b_{s,i}$, the value of an input parameter $p \in b_{s,i}^{\text{in}}$ is denominated $v_s^{\text{in}}(i, p) \in p^{\text{type}}$. In analogy, output parameter values are labelled $v_s^{\text{out}}(i, p)$ for $p \in b_{s,i}^{\text{out}}$. Thus, v_s^{in} and v_s^{out} model a lookup that maps each parameter of each skill graph node to its current value. Values can be input by the user during task modelling. They may also emerge as a consequence of data flow through actions of a skill at execution time in case of consecutive part manipulation (cf. Section 3.1.2). Consistent data flow is supported by the prediction function b^π . This function serves the purpose of forecasting an output parameter value for each input object provided by some parameter $p \in b^{\text{in}}$ with $p^{\text{type}} = \bar{D}$ after successful execution of an instance of template b . From a pragmatic point of view, prediction functions can be seen as yet another implementation of a symbolic action. This implementation does however not control hardware, but rather simulates action execution. Let $\bar{b}^{\text{in}} \subseteq b^{\text{in}}$ denote the set of all object-related input parameters of b , with

$$\bar{b}^{\text{in}} = \{p \in b^{\text{in}} \mid p^{\text{type}} = \bar{D}\} = \{\bar{p}_1^{\text{in}}, \bar{p}_2^{\text{in}}, \dots, \bar{p}_{|\bar{b}^{\text{in}}|}^{\text{in}}\}. \quad (3.5)$$

The value of each element in \bar{b}^{in} must be mapped to an output parameter describing the object state after execution. We therefore demand that b^{out} have a subset $\bar{b}^{\text{out}} \subseteq b^{\text{out}}$ of corresponding output parameters matching the object-related input, where

$$\bar{b}^{\text{out}} = \{\bar{p}_1^{\text{out}}, \bar{p}_2^{\text{out}}, \dots, \bar{p}_{|\bar{b}^{\text{in}}|}^{\text{out}}\}, \quad \bar{p}_j^{\text{out}} = \bar{p}_j^{\text{in}}, \quad j \in \{1, \dots, |\bar{b}^{\text{in}}|\}. \quad (3.6)$$

With these sets defined, the prediction function $b^\pi : \bar{b}^{\text{in}} \times \bar{D} \rightarrow \bar{D}$ of each basic action needs to be implemented to transform input objects into their state after execution. Then, each action instance $b_{s,i}$ within a skill can fill output parameters with values describing action effects on parts, with

$$v_s^{\text{out}}(i, \bar{p}_j^{\text{out}}) = b_{s,i}^\pi(\bar{p}_j^{\text{in}}, v_s^{\text{in}}(i, \bar{p}_j^{\text{in}})), \quad \bar{p}_j^{\text{out}} \in \bar{b}_{s,i}^{\text{out}}, \bar{p}_j^{\text{in}} \in \bar{b}_{s,i}^{\text{in}}. \quad (3.7)$$

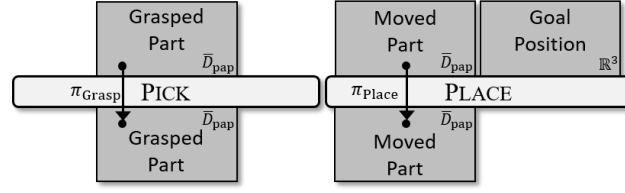
The above constraints force each input object to be transformed and forwarded into the basic action output. On that basis, a skill graph structure with a similar property can be defined in the next section. This renders automated prediction of effects for arbitrary skills composed of several basic actions feasible.

Two example basic actions for use with the parts domain \bar{D}_{pap} are visualized by Figure 3.3. Their semantics may e.g. be defined as follows: The PICK action b_{PICK} with

$$b_{\text{PICK}} = (\{(\text{GraspedPart}, \bar{D}_{\text{pap}})\}, \{(\text{GraspedPart}, \bar{D}_{\text{pap}})\}, \pi_{\text{PICK}}) \quad (3.8)$$

is intended to pick up an object of a certain type. It therefore takes an object state as input. Any implementation is assumed to calculate a suitable grasping pose based on the object type and position. This pose is approached from a dedicated transfer level, and the gripper is closed. If closing the gripper results in the calculated finger positions grasping is assumed successful. A detach-motion is carried out. Otherwise, the part

Figure 3.3.: Example basic actions PICK and PLACE (light grey) take object states and target positions as parameters (dark grey). Corresponding input and output parameters are linked by prediction functions (black arrows).



provided at runtime did not match the type specified by the parameter – the gripper is opened, and execution results in the failure state after detaching to the transfer level. PLACE has two input parameters: The state of an object to be placed implies that some part is already held in the gripper. In addition, the centre of gravity coordinates of this object after placement are given by the goal position. The tuple for b_{PLACE} is

$$b_{\text{PLACE}} = (\{(\text{MovedPart}, \bar{D}_{\text{pap}} \}, \{ \text{GoalPosition}, \mathbb{R}^3 \} \}, \{ \text{MovedPart}, \bar{D}_{\text{pap}} \}, \pi_{\text{PLACE}}). \quad (3.9)$$

Placing is achieved by a point-to-point motion from the current robot position. If the target position is free, the part can be moved there. The gripper is then opened, and the robot can move back to the transfer level. This procedure may fail when the target position is already occupied by another object. In this case, the approach motion will eventually be stopped, and execution failure is reported.

The prediction functions π_{PICK} and π_{PLACE} link input object-related parameters to their state in the case of action success. After execution, PICK will provide an object state describing the part with its position on the transfer level. When combining several actions, this output state can e.g. be used as an input to the ‘Moved Part’-parameter of PLACE. This action in turn calculates the object state after placing the part at the specified target position. Details on the graph structure to link actions in terms of control and data flow are given in the next section.

3.1.2. Skill Graph Structure

The general structure and semantics of skills has already been introduced by Figure 3.2. Particularly, skill control flow is designed to start from a designated node and to exit with one out of three states (SUCCESS, ABORT, or un-handled ERROR). The following *control nodes* are introduced to model graphs according to this structure:

- START marks the entry point of a skill.
- SUCCESS represents the state that a skill enters upon successful execution.
- Skills terminate with ABORT if an error handled by reverse execution occurred.
- If an error that cannot be handled by reverse execution is encountered skill execution terminates in ERROR.

These nodes can be seen as basic action tuples with empty input and output parameter sets, and thus without an effect b_π on the physical world. Extending Equation 3.1, each

skill s is then fully specified by a graph with control flow s^{flow} , data flow s^{data} and a node list s^{B} with the following properties:

$$\begin{aligned} s &= (s^{\text{B}}, s^{\text{flow}}, s^{\text{data}}) \\ s^{\text{B}} &= (b_{s,1}, b_{s,2}, \dots, b_{s,|s^{\text{B}}|}) \\ b_{s,1} &= \text{START}, \quad b_{s,2} = \text{SUCCESS}, \quad b_{s,3} = \text{ABORT}, \quad b_{s,4} = \text{ERROR} \end{aligned} \quad (3.10)$$

The node list must at least contain the four unique control nodes. All other nodes correspond to one of the available basic action templates, i.e. $b_{s,5} \dots b_{s,|s^{\text{B}}|} \in B$. Any basic action can of course be instantiated multiple times within a single skill.

Control flow is defined by a set of so-called *flow connections*. Each of these connections $\varphi \in s^{\text{flow}}$ is itself a tuple

$$\begin{aligned} \varphi &= (i, j, k), \\ i, j, k &\in \{1, \dots, |s^{\text{B}}|\}, \quad i \neq j \neq k \end{aligned} \quad (3.11)$$

consisting of three node indices i, j and k . A connection φ defines two outgoing, directed edges starting in $b_{s,i}$: If the basic action underlying $b_{s,i}$ is performed successfully during execution, then the next action along the control flow is $b_{s,j}$. Otherwise, the skill execution engine will transition to $b_{s,k}$ as the next action. The successful execution path (Figure 3.2, green) can then be constructed by tracing the edges indicating success. We will refer to this path as $s_+^{\text{B}} = (b_{s,1}, \dots, b_{s,m}, \dots, b_{s,2})$. With Equation 3.10 s_+^{B} starts at $b_{s,1} = \text{START}$ and ends in $b_{s,2} = \text{SUCCESS}$. The other success path nodes $b_{s,m}$ ($m \in \{5, 6, \dots, |s^{\text{B}}|\}$) are sorted in the order of their occurrence along the path. Only graphs without cycles are considered in this work to ensure that s_+^{B} is clearly defined for each possible skill – this boundary condition must be taken into account when modelling a new skill.

In contrast to flow connections, a *data connection* $\delta \in s^{\text{data}}$ represents a single directed edge that transforms an action output parameter into an input parameter of the action at the other end. Each data connection δ is a tuple

$$\begin{aligned} \delta &= (i, j, \delta^{\text{out}}, \delta^{\text{in}}) \\ i, j &\in \{1, \dots, |s^{\text{B}}|\}, \quad i \neq j, \quad \delta^{\text{out}} \in b_{s,i}^{\text{out}}, \quad \delta^{\text{in}} \in b_{s,j}^{\text{in}} \end{aligned} \quad (3.12)$$

saying that the output parameter δ^{out} of action $b_{s,i}$ is connected to the input parameter δ^{in} of action $b_{s,j}$. When the control flow transitions from $b_{s,i}$ to $b_{s,j}$, the value $v_s^{\text{out}}(i, \delta^{\text{out}})$ as produced by $b_{s,i}$ is thereby passed on to $b_{s,j}$, i.e. $v_s^{\text{in}}(j, \delta^{\text{in}}) = v_s^{\text{out}}(i, \delta^{\text{out}})$. This may e.g. be used to fill parameters with data gathered by sensor actions on the one hand. The mechanism is yet particularly important for propagating object-related input parameters throughout the graph to determine the expected state of objects after consecutive manipulations. Starting from the START node, action effects can be calculated and forwarded to the next action along s_+^{B} based on the prediction functions b^π . Object states reflecting skill outcomes can then be collected from skill output parameters.

The skill output parameters s^{out} as well as the input parameters s^{in} that the user needs to specify when parametrising a skill can be deduced automatically from the graph structure: Each outgoing data connection says that information carried by the output

parameter will be processed further by some subsequent action. By implication, values of parameters without an outgoing edge to another node along s_+^B are final – they must belong to the output after successful skill execution. Collecting parameters with this property from all actions along the successful execution path s_+^B thus leads to a set s^{out} of *parameter instances* (m, p) putting parameter p into the context of a graph node with index m , i.e. the skill output is given by

$$\forall b_{s,m} \in s_+^B \quad \forall p \in b_{s,m}^{\text{out}} : (m, p) \in s^{\text{out}} \Leftrightarrow \left(\nexists (i, j, \delta^{\text{out}}, \delta^{\text{in}}) \in s^{\text{data}} : i = m \wedge \delta^{\text{out}} = p \wedge b_{s,j} \in s_+^B \right). \quad (3.13)$$

Here, only actions on the path of successful execution are considered. Any paths for error handling are omitted by purpose as they are not informative about the intended target state after applying a skill to certain parts.

By contrast, all graph nodes must be considered when deriving skill input parameters. In analogy to s^{out} , we can define s^{in} to denominate all input parameters of all basic actions with no entering data flow edge. Values for these parameters cannot emerge from data flow – they must therefore be set by the user during skill parametrisation. Only then is the skill fully parametrised and ready for execution. The set of parameter instances to be provided by the user is thus implicitly given by

$$\forall b_{s,n} \in s^B \quad \forall p \in b_{s,n}^{\text{in}} : (n, p) \in s^{\text{in}} \Leftrightarrow \left(\nexists (i, j, \delta^{\text{out}}, \delta^{\text{in}}) \in s^{\text{data}} : j = n \wedge \delta^{\text{in}} = p \right). \quad (3.14)$$

Figure 3.4 shows an example of a simple skill graph matching the above formalism. The skill uses the basic actions PICK and PLACE from Figure 3.3 and thereby realises an object transfer. Parts are specified by their type and position in the example domain \bar{D}_{pap} (cf. Section 3.1.1). The skill has nodes $s^B = (1, 2, 3, 4, 5, 6, 7)$ – for a more compact representation, the full tuples specifying PICK and PLACE according to Equations 3.8 and 3.9 are here replaced with these identifiers as assigned in Figure 3.4. The nodes $s_+^B = (1, 5, 6, 2)$ lie on the success path. When the SUCCESS node is entered, the part has previously been grasped and relocated to the specified target position (Actions 5 and 6). The shown control flow furthermore handles the error case in which the goal position is already blocked by some other part. In this case, the PLACE Action 7 tries to return the object to its prior position as stored in the initial input state to Action 5. It is important to notice that object states are seen as *composite parameters* in the skill framework implementation. This way, aspects of parts can be addressed and passed on individually by data connections. E.g., the position of the input object state serves as the goal position of the error handling PLACE Action 7. Any parameter without an incoming data edge is part of the skill input, i.e. $s^{\text{in}} = \{(5, \text{GraspedPart}), (6, \text{TargetPosition})\}$. The MovedPart output parameter of Action 6 is identified by the pair $(6, \text{MovedPart})$. This parameter instance matches the condition of Equation 3.13 as the outgoing data connection $\delta = (6, 7, \text{MovedPart}, \text{MovedPart})$ leaves the successful execution path. The instance is thus the sole output parameter so that $s^{\text{out}} = \{(6, \text{MovedPart})\}$. More complex skills utilizing the full spectrum of motor, perception and communication actions are introduced in the next section where a concrete domain is fully specified.

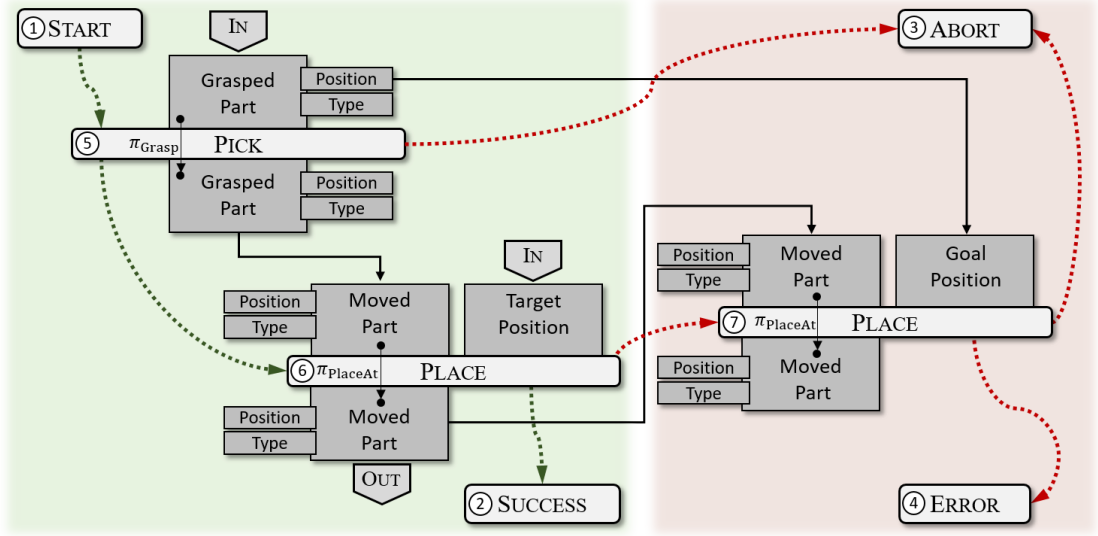


Figure 3.4.: An example skill graph uses PICK and PLACE motor actions (light grey) to realise a simplified pick-and-place operation. Input parameters (IN) specify the affected object and target position. Data flow (solid edges) leads to an output parameter (OUT) containing the object state after execution, if the control flow (dashed edges) follows the successful execution path (green area and arrows). The error case where the target position is blocked is handled by reverse execution (red area and arrows).

3.1.3. Benchmark Domain

The previous sections have introduced the concept of skill graphs along with rather abstract, basic examples. Let us now look at a concrete domain in line with this formal framework. This domain is designed for demonstrating and benchmarking the flexible teaming approach proposed in this thesis. Therefore, two main criteria need to be met: (i) The domain should cover skills that are typically used in industrial applications to the greatest possible extent. (ii) Skills should enable composition of tasks that are scalable, e.g. in their overall duration, the degree of close human-robot interaction etc. Aside from these functional requirements, domain complexity is furthermore limited as follows: The set of used parts prescribes the effort that must be put into object recognition, grasp planning etc. These problems lie out of the scope of this thesis – therefore, only parts with low geometric complexity and clear features for identification are used. Similarly, motions happen on a transfer level as far as possible to avoid collisions with the environment without explicit motion planning. These simplifications ease prototype implementation but are not inherent limitations of the conceptual framework.

Objects: Figure 3.5 shows examples of the benchmark domain objects. A surface model is assumed given for each object type. There are overall seven part types

$$\mathcal{T} = \{\text{RedBlock}, \text{WhiteBlock}, \text{RedLatched}, \text{WhiteLatched}, \text{YellowBase}, \text{Tool}, \text{Container}\}. \quad (3.15)$$

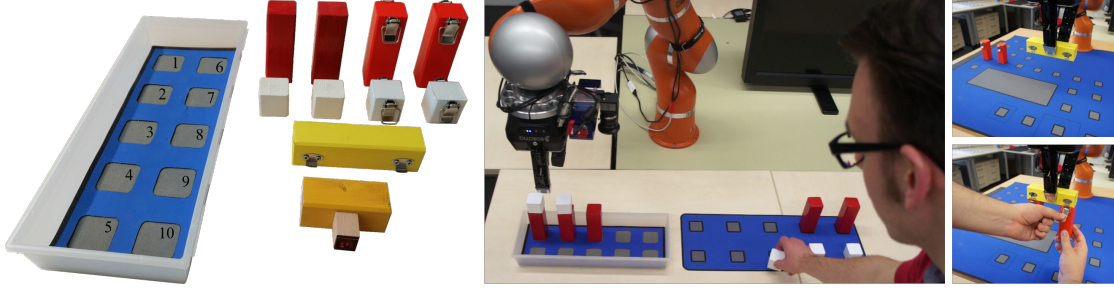


Figure 3.5.: The parts of the benchmark domain are clearly distinguished by their size and predominant colour (left). The main focus lies on pick-and-place tasks, e.g. palletising of objects into containers (middle). Parts with hooks and latches emulate basic assembly steps (right).

The rectangularly shaped blocks differ strongly in colour and dimensions. Red and white blocks may be equipped with latches. These latches match hooks attached to the yellow base parts, thus supporting basic workpiece mating steps. A smaller yellow block with a stamp attached represents a tool for applying labels (e.g. adhesive barcode stickers) to other parts. Containers define *slots*, i.e. numbered positions within the container. Each slot is characterized by its relative position within the coordinate frame of the container surface model. Instances of these part types are defined by their aspects ‘position’ and ‘type’, and by a boolean flag indicating whether they were marked with the tool – object states are hence specified by an element of $\bar{D}_{\text{bench}} = \mathcal{T} \times \mathbb{R}^3 \times \{\text{TRUE}, \text{FALSE}\}$. All parts are easy to acquire. This supports reproducible and comparable experiments.

Basic Actions: Picking, placing and holding parts are frequent robot actions in recent human-robot collaborative studies with applications to assembly and manufacturing [130]. These actions therefore form the backbone of B^{motor} in the benchmark domain. The motor action template set is composed of

$$\begin{aligned}
 B^{\text{motor}} = \{ & \text{PICK}, \text{PLACE}, \text{PLACEONOBJECT}, \text{PLACEINSLOT}, \text{TRANSFEROVERSLOT}, \\
 & \text{TRANSFEROVER}, \text{TRANSFEROVEROBJECT}, \text{MOVEOVEROBJECT}, \\
 & \text{HOLDFORASSEMBLY}, \text{PLACEASSEMBLY}, \text{SHAKE}, \text{APPLYTOOL} \}.
 \end{aligned}
 \tag{3.16}$$

The input to each of these actions is summarised by Table 3.1. Parameter values are taken from the domains $PD_{\text{bench}} = \{\bar{D}_{\text{bench}}, \mathbb{R}^3, \mathbb{N}_0^+\}$. The semantics and definition of PICK and PLACE have already been introduced by Figure 3.3 in Section 3.1.1. Additional actions PLACEONOBJECT and PLACEINSLOT behave similarly but accept different parameters. E.g., PLACEONOBJECT targets stacking of parts by taking two object states as input. One of them describes the part to be moved actively and the other specifies the bottom object in the stack yet to emerge. With PLACEINSLOT, a part goal location is determined by a target container and a number to identify a position in its slot grid. These parameter variations model cases, where different calculations are needed to map task-level parameters to concrete robot commands with similar overall effects. As a result, basic action variants liberate the user e.g. from calculating the absolute goal

Basic Action	Input Parameter (Type)
PICK	Object to grasp (\bar{D}_{bench})
PLACE	Grasped object (\bar{D}_{bench}), target position (\mathbb{R}^3)
PLACEONOBJECT	Grasped object (\bar{D}_{bench}), target object (\bar{D}_{bench})
PLACEINSLOT	Grasped object (\bar{D}_{bench}), container (\bar{D}_{bench}), slot ID (\mathbb{N}_0^+)
TRANSFEROVER	Grasped object (\bar{D}_{bench}), target position (\mathbb{R}^3)
TRANSFEROVEROBJECT	Grasped object (\bar{D}_{bench}), target object (\bar{D}_{bench})
TRANSFEROVERSLOT	Grasped object (\bar{D}_{bench}), container (\bar{D}_{bench}), slot ID (\mathbb{N}_0^+)
MOVEOVEROBJECT	Target object (\bar{D}_{bench})
HOLDFORASSEMBLY	Base part (\bar{D}_{bench}), component A (\bar{D}_{bench}), component B (\bar{D}_{bench})
PLACEASSEMBLY	Base part (\bar{D}_{bench}), component A (\bar{D}_{bench}), component B (\bar{D}_{bench})
SHAKE	Grasped object (\bar{D}_{bench}), number of repetitions (\mathbb{N}_0^+)
APPLYTOOL	Grasped tool (\bar{D}_{bench}), part to apply a label to (\bar{D}_{bench})

Table 3.1.: Input parameters of basic motor actions B^{motor} in the benchmark domain

position of a part when placed in a certain container slot. They create the object-centric interface for convenient, task-level skill parametrisation.

In addition to picking and placing, B^{motor} offers robot motion on a predefined transfer level. They enable collision-avoiding part transfers without dedicated motion planning. The variants of TRANSFER have the same input as the PLACE variations. In analogy, they place the robot hand with an object grasped above some given position, another object or a container slot. MOVEOVEROBJECT targets a transfer level motion over a part like TRANSFEROVEROBJECT, but without an object grasped. We will assume position-controlled, non-compliant motions for now – the TRANSFER and MOVE actions will thus always succeed, whereas placing may fail if the target location is blocked.

The remaining actions are more domain-specific. HOLDFORASSEMBLY models the robot role in the process where a yellow base part is mated with two latched components A and B (Figure 3.5, right). This action brings along an additional PLACEASSEMBLY taking all assembled parts as input as the other PLACE actions can only process one workpiece. The template SHAKE emulates part processing by applying a given number of shaking motions to a grasped object to emulate e.g. mixing of fluids in a bottle. Similarly, APPLYTOOL processes a part by applying the previously grasped tool.

The basic action set is completed by two sensor actions and a communication action:

$$\begin{aligned}
 B^{\text{sense}} &= \{\text{CAPTUREIMAGE}, \text{CHECKLABEL}\} \\
 B^{\text{comm}} &= \{\text{QUERYCOMPLETION}\}
 \end{aligned}
 \tag{3.17}$$

The robot-mounted camera is triggered with CAPTUREIMAGE. The resulting image is then provided as an action output parameter value. CHECKLABEL takes an object state from \bar{D}_{bench} as input. This action centres the camera above the specified part to check whether the tool has previously been applied. If the part does not have a label attached, CHECKLABEL indicates successful execution. An implementation of QUERYCOMPLETION is intended to open a communication channel and ask human partners whether they completed their role within a collaborative skill. Communication is blocking and ends as

soon as an answer was provided. Execution of a `QUERYCOMPLETION` implementation is successful if the communication partner confirms task completion positively.

Skills: With these basic action templates at hand, the skills listed in Table 3.2 can be composed. The table summarises input parameters, effects and the sequence of basic actions along the successful execution path for each skill. In analogy to the `PLACE` and `TRANSFER` action template variations (Table 3.1), `Pick&Place`, `Pick&PlaceOnTop` and the `Pick&PlaceInContainer` skill enable transferring parts with different task-level input parameters. Pick-and-place skills are suited for scaling the overall number of parts used in a task with several operations. In consequence, overall task duration and space usage for storing parts in the workspace can be varied across a set of benchmark tasks. `ShakeAndTransfer` reuses the structure of `TransferToContainer` but adds a `SHAKE` action after grasping the part. Operations with different individual durations can be introduced into a task with the number of shaking repetitions. Scalability regarding the degree of direct interaction is integrated into the benchmark domain by the `Assemble` skill. The graph defining this skill encodes the robot roll in the mating operation (Figure 3.5, right). After grasping the base part (`MOVEOVER` and `PICK`), it is transferred to and presented at a predefined location for assembly (`HOLDFORASSEMBLY`). The subsequent `QUERYCOMPLETION` action makes the robot wait for another agent to confirm latching of the two other components. A weaker form of synchronization by sharing a tool can be added using the `ApplyLabel` skill. It widely resembles picking and placing of parts: A tool is first picked, moved over and pressed onto (`APPLYTOOL`) the part to mark. The tool is then transferred over and finally placed at its initial pose.

Errors are handled similarly in all graphs of the above skills. There are two general error cases for picking and placing: (i) A part could not be grasped at its expected position, or (ii) placing failed as the goal position was blocked. No error handling is needed in the first case. The world state has not yet been changed – skill execution can thus transition into the `Abort` terminal node without further action after re-opening the gripper. Reverse execution is only required in the latter case in which a part has already been picked up and moved. In this case, the robot will try to return it to the initial location. If this fails, the skill exits in the `ERROR` state. Otherwise the final state is `ABORT`. The `ApplyLabel` skill introduces another error case in addition to situations, where the tool could not be grasped – an intrinsically robust realisation must furthermore consider situations, where the robot would erroneously apply a duplicate marking to a part that has the required barcode sticker already attached. To this end, `ApplyLabel` will return the tool and exit with `ABORT` if the `CHECKLABEL` action indicates failure.

The proposed human-robot teaming framework does not only rely on the skill framework for task modelling and execution. Additional *utility skills* are also used by reasoning components, e.g. to issue active perception. They are not available as a part of tasks during graphical modelling. `Approach&CaptureImage` is an example for this sort of skills: When supplied with an object to observe, this skill will place the eye-in-hand camera above it. Then, the `CAPTUREIMAGE` action triggers the camera to generate the requested sensor data. Making all components control sensors and actuators via the skill execution engine renders the overall system hardware-independent.

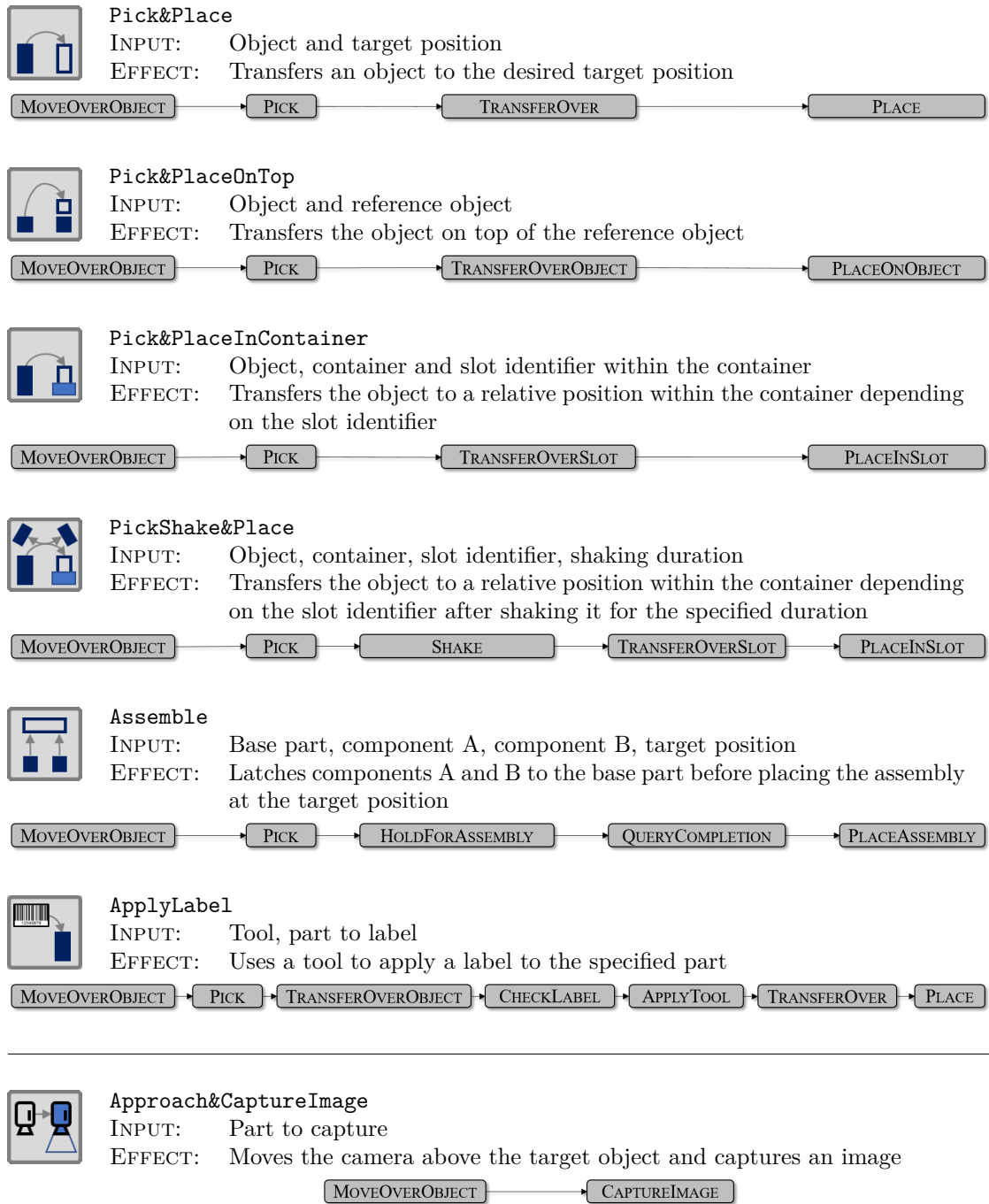


Table 3.2.: Skills of the benchmark domain for human-robot teams with their input, effects and basic actions along the successful execution path (grey box sequences)

3.2. Shared Task Model Generation

Skills provide a symbolic and parametric representation of process steps. The special graph structure underlying these process step templates provides necessary input parameters to be specified. We will call the combination of a skill with values for each of its input parameters an *operation*. For investigating flexible teaming in larger tasks, operations need to be combined into task models that all involved agents know. The proposed concept for establishing such shared task mental models has two stages: (i) Procedural knowledge on the task is first queried from the user with a graphical user interface (Section 3.2.1). A functionally equivalent tool for the benchmark domain has been evaluated in the author’s prior work [133] – this section builds upon this work by rendering the task editor more generic and domain-independent. In particular, the set of skills offered to the user can conveniently be extended or changed using mechanisms of the skill framework. (ii) The resulting precedence graph of operations is then automatically annotated with operation pre- and postconditions. Conditions enrich the task model with the information needed by robots to observe task progress (Section 3.2.2).

3.2.1. Graphical Modelling of Precedence Graphs

Tasks with parallelism are often represented by precedence graphs [54] in industrial domains. Formally, this structure models tasks as partially ordered sets of operation nodes. Let T denote the set of nodes, with

$$T = \{\tau_{\text{start}}, \tau_1, \dots, \tau_N, \tau_{\text{end}}\}. \quad (3.18)$$

Each of the N nodes τ_1, \dots, τ_N represents an operation, i.e. in our case a skill paired with values for all necessary input parameters. We will use s_τ when referring to the skill that an operation $\tau \in \{\tau_1, \dots, \tau_N\}$ instantiates. Let furthermore $<_T$ denote a partial order on T . This order says that τ_i must be done before τ_j , if and only if $\tau_i <_T \tau_j$ ($i \neq j$). We say that there is a precedence relation between τ_i and τ_j if there is no other operation τ_k to be done chronologically ‘in between’ τ_i and τ_j ($\nexists \tau_k : \tau_i <_T \tau_k <_T \tau_j \wedge i \neq k \neq j \wedge i \neq j$). A task is then visualized by a precedence graph as follows: A directed edge leaves τ_i and enters τ_j provided there is a precedence relation between these nodes. To connect the graph, each node without an incoming precedence edge is linked to the dedicated *start node* τ_{start} by an additional incoming edge. Accordingly, nodes without an outgoing precedence relation are connected to the *end node* τ_{end} . Valid precedence graphs are assumed to be connected and free of cycles. Furthermore, a path must exist from τ_{start} to each τ_k and from each τ_k to τ_{end} ($k = 1 \dots N$). Creating a task model $(T, <_T)$ can thus be decomposed into two sub-problems to be solved by graphical programming: First, skills need to be parametrised by specifying their input parameters. The same skill may, of course, serve as a template for several operations with differing input – the task model therefore holds *operation input values* $v_\tau^{\text{in}}(m, p)$ for each parameter instance $(m, p) \in s_\tau^{\text{in}}$ of skill s_τ (Equation 3.14) in the context of operation τ . The resulting operations can then be linked with precedence relations to express the order of operations in the process.

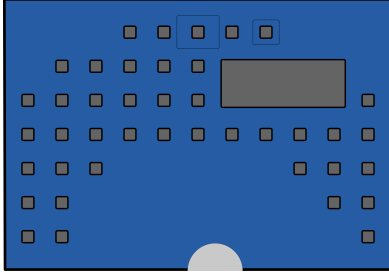


Figure 3.6.: Workspace layouts are an optional input to the task editor. They ease inputting part positions (dark grey) and transferring the modelled workspace setup to the physical world around the robot base position (light grey).

Graphical robot programming is a well-studied topic [104]. Creating precedence graphs for flexible teams is however not robot programming in the strict sense – the model is not limited to robot commands or actions, but rather covers all operations of a process. These operations need not necessarily be feasible for the robot, i.e. the task model may potentially cover commands to several agents. Still, robot programming provides reasonable starting points for establishing shared task models. In particular, this work transfers ideas from *icon-based* and *CAD-based* programming to human-robot teaming. Icon-based approaches build flowchart-like representations of robot programs by connecting function blocks, each represented by a pictogram [17]. Modelling precedence graphs similarly requires a mode to connect operations, and skills are a direct source for icons (Table 3.2). By contrast, recent CAD-based approaches enable task-level programming by virtual manipulation of parts [66]. Due to the equally clear focus on parts as input parameters to skills, this paradigm is suitable to support skill parametrisation in a virtual representation of the workspace.

The task editor relies on three sources of input: (i) A set of skills, represented by extended markup language representations of their graph structure, is provided at program start-up. (ii) The skill execution engine knows basic actions and especially implements their prediction functions. The engine can thus simulate skill execution for a set of input parameters, thus supporting visualization of effects in the editor. (iii) Optionally, a *workspace layout* can be specified (Figure 3.6). Conceptually, these layouts define small, arbitrarily shaped areas called *slots* in analogy to the containers of the benchmark domain. Layouts are provided as vector graphics files. The editor can automatically extract the locations of all slots from these files. Offering slots to the user renders selection of positions in the workspace more convenient. Furthermore, all slots are given in the world coordinate frame of the robot, i.e. they are specified relatively to the robot base segment. Affixing an identical, printed version of the layout in the physical workspace enables straightforward transfer of physical parts to the positions as modelled in the virtual environment (Section 6.1.1).

Based on these input components, the graphical user interface (GUI) unites icon-based modelling with CAD-based elements for skill parametrisation as shown in Figure 3.7. Available parts and skills are offered in drop-down menus. Instances of parts are created by specifying their initial position in a dialogue window. A position can be chosen by clicking a slot of the workspace layout instead of manually adjusting the part coordinates. The editor will then calculate the part pose based on the slot position and part extents

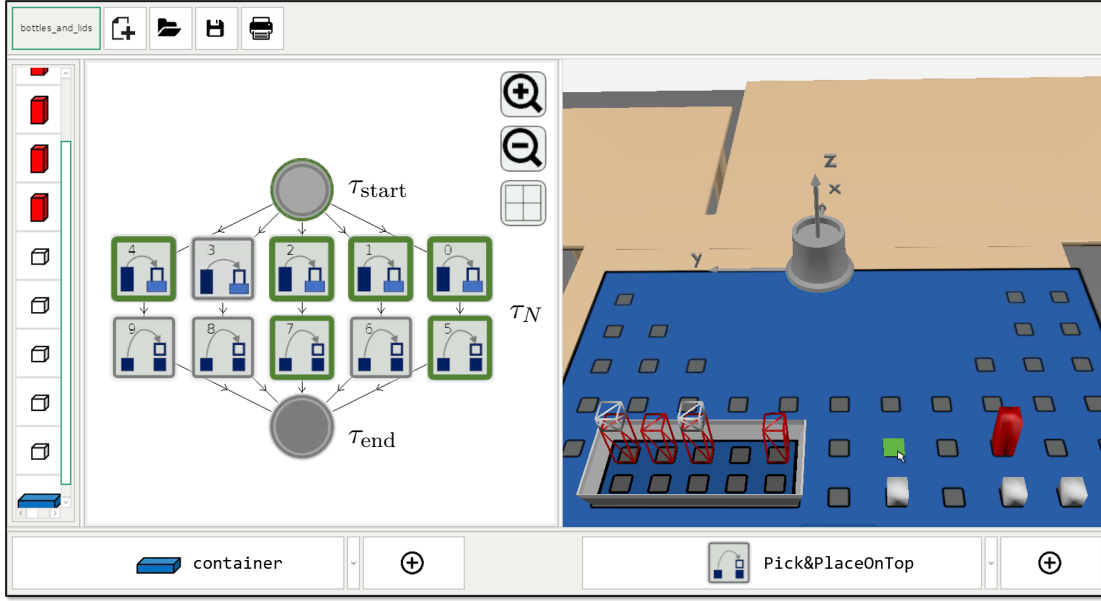


Figure 3.7.: The task editor combines CAD-based skill parametrization via a virtual workspace representation (right) with icon-based modelling of precedence graph structures (centre) and a parts list (left). Parts and skills are instantiated via drop-down boxes (bottom) and represented as icons or solid objects in the virtual world. When clicking an operation, its effects are visualized by wireframes of the expected object states after skill execution.

extracted from the part surface model (Section 3.1.1). For better usability, slots are highlighted green when hovering the mouse pointer over them. Any inserted parts appear in a parts list and in the virtual workspace.

When instantiating a skill, the user is prompted with a window for parametrization. These windows are auto-generated: By construction of the skill graphs, parameters can only be taken from the set of parameter domains PD . This set is finite and fixed for a domain. We can thus assign a suitable GUI control type to each parameter type. A parametrization dialogue can then be assembled for an arbitrary skill s by adding matching controls for each of the input parameter instances s^{in} (Equation 3.14). Examples from the benchmark domain are shown in Figure 3.8. The `Pick&PlaceInContainer` skill requires the affected object, a container instance and the slot identifier within this container as input. These parameters are queried by one tab-page each. Object state parameters from the domain \bar{D}_{bench} are specified by selecting one of the objects in the parts list, presented in a drop-down list. A similar list is filled with the available slot identifiers of the selected container. The part goal position from \mathbb{R}^3 can be adjusted with a slider control for each coordinate value. Skill parametrization is also supported by the virtual environment: In addition to a suitable GUI control element, each parameter type is linked to a callback function. On clicking into the environment, the function matching the parameter type associated to the active tab-page is triggered. Similar to the parametrization of parts, positions and object states can this way be selected by clicking

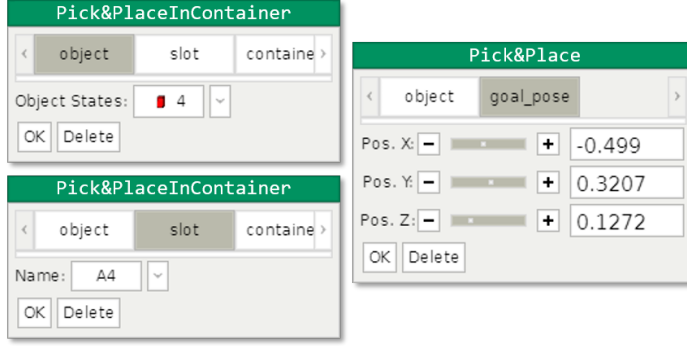


Figure 3.8.: Skill input parameters are queried from the user with different auto-generated GUI elements linked to individual parameter types.

slots or objects in the virtual workspace. The handler function then maps the selected parts or locations to parameter values. Extending the editor towards new parameter domains thus means specifying a GUI control covering this domain and implementing a handler for manipulations in the virtual workspace.

Operations are represented by pictograms in the icon-based editor component (Figure 3.7, centre) after initial skill parametrisation. The parametrisation window can be accessed for later adjustment of input values or for deleting an operation via these icons. After inserting several operations, precedence relations can be established among them and the dedicated circular icons of the start node τ_{start} and the end node τ_{end} . This is realised by mouse drag&drop motions, starting from some node and terminating at the chronological successor of this node according to the intended task structure. Upon clicking the icon associated with some operation τ_i it is highlighted with a green border. The effects of τ_i and all necessary preceding operations τ_j with $\tau_j <_T \tau_i$ are displayed as additional, wireframe-rendered objects in the virtual workspace. Parts representing operation effects are in turn clickable and may serve as skill input parameters, e.g. when specifying the reference object of a `Pick&PlaceOnTop` skill. Modelling consecutive part manipulations is facilitated this way.

3.2.2. Annotation with Operation Pre- and Postconditions

Visualization of operation effects and automatic adaption of the task editor GUI to arbitrary skills are enabled by the special skill graph structure outlined in Section 3.1.2. The mechanisms supporting these features equally provide means to deduce operation pre- and postconditions. These conditions allow robots to understand task progress. Let $E_t \subset \bar{D}$ denote a set of objects that the robot system has previously sensed. This *world model* encodes robot knowledge on parts in the workspace. An operation can only be executed successfully if all required parts are available. The parts needed can be deduced from skill input parameters: Let τ be an instance of skill s_τ with input parameter instances s_τ^{in} . Building upon Equation 3.14, the set $\bar{s}_\tau^{\text{in}} \subseteq s_\tau^{\text{in}}$ with

$$\bar{s}_\tau^{\text{in}} = \{(m, p) \in s_\tau^{\text{in}} \mid p^{\text{type}} = \bar{D}\} \quad (3.19)$$

contains all input parameters referring to a part processed by τ . The user is prompted to specify all values $v_\tau^{\text{in}}(m, p)$ ($(m, p) \in s_\tau^{\text{in}}$) during task modelling to form an operation that instantiates s_τ . The gap between expected parts according to the task model and physically available objects in the workspace can then be bridged by demanding

$$\forall (m, p) \in \bar{s}_\tau^{\text{in}} : (\exists e_p \in E_t : e_p \approx_{\bar{D}} v_\tau^{\text{in}}(m, p)) \quad (3.20)$$

before starting execution of an instance of s_τ . The system can thus avoid skill execution failure by checking preconditions according to Equation 3.20 against previously gathered sensor data stored in E_t . If this check fails, some other agent may have claimed the parts to carry out the operation.

Progress achieved by other agents can be extracted from sensor data analogously. The skill framework is designed to predict expected skill outcomes. As with the input parameters, there is a subset $\bar{s}_\tau^{\text{out}} \subset s_\tau^{\text{out}}$ of object-related skill output parameters. In analogy to v_τ^{in} , a lookup for skill output parameter values is denoted by $v_\tau^{\text{out}}(m, p)$ for parameter instances $(m, p) \in \bar{s}_\tau^{\text{out}}$ – the system calculates respective values by propagating input parameters along the successful execution path and applying basic action prediction functions repeatedly. An operation is assumed done if and only if the postconditions

$$\forall (m, p) \in \bar{s}_\tau^{\text{out}} : (\exists e_p \in E_t : e_p \approx_{\bar{D}} v_\tau^{\text{out}}(m, p)) \quad (3.21)$$

are true. Facts about satisfied postconditions are particularly important for maintaining an estimate of task progress as introduced in the following section.

3.3. Task Execution Principle

A task model emerging from the procedure described in Section 3.2 provides sufficient information for a robot to participate in the task. In teaming scenarios, participation inevitably presupposes an understanding of task progress induced by one's partners. To this end, the robot system must maintain an estimate P_t of task progress at time t . This estimate stores one out of the three states INACTIVE, ACTIVE and DONE for each operation τ of a task $(T, <_T)$, i.e.

$$P_t : T \rightarrow \{\text{INACTIVE}, \text{ACTIVE}, \text{DONE}\}. \quad (3.22)$$

Initially, P_t is only ACTIVE for direct successors of the start node τ_{start} in the precedence graph of T . All other operations are tagged INACTIVE. The system will make repeated observations of the workspace while task execution unfolds (Chapter 5). From the moment the system detects that all postconditions of some operation τ' are satisfied in the world model, $P_t(\tau')$ is DONE. This holds, of course, also for situations where the robot itself has carried out τ' successfully via the skill execution engine. Precedence graphs capture earlier-later-relations among operations. This means that any preceding operation on any path between τ' and τ_{start} is a requirement for doing τ' . We furthermore assume cooperative workers who always follow the task model correctly (Section 1.2). Under this assumption, all predecessors τ'' of τ' , where $\tau'' <_T \tau'$, can also be marked DONE

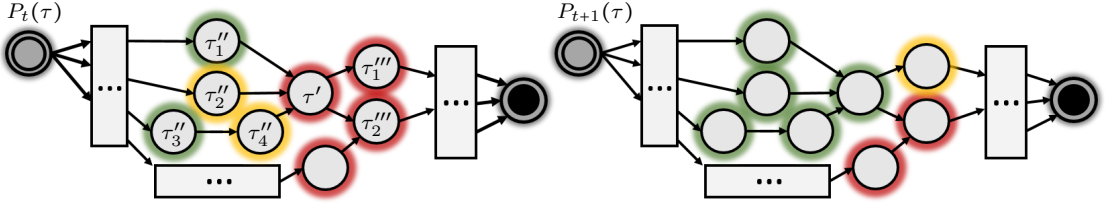


Figure 3.9.: The system maintains an estimate of task progress (left). After successful execution, operations are flagged DONE (green). ACTIVE operations (yellow) are up next. An operation is INACTIVE, if there are predecessors that have not yet been DONE. Fulfilled postconditions indicate operation success. They trigger an update (right) of the initial estimate (left).

as soon as τ' is DONE – these operations must also have been completed. Figure 3.9 shows an example of this update step. The initial estimate is advanced by refreshing the values of τ' , τ_2'' and τ_4'' that are newly marked DONE. By implication, a newly DONE operation means that the prerequisites of subsequent operations τ''' may now be met. If so, then these successor nodes may be marked ACTIVE. In the example of Figure 3.9, this applies to τ_1''' – by contrast, τ_2''' has another direct predecessor in addition to τ' . This predecessor is not yet DONE and prevents τ_2''' from being activated.

With the task progress estimate P_t , the basic task execution principle is as follows: The system first needs to extract all operations that are ACTIVE at the moment. These operations are candidates for execution provided that their sensory preconditions according to Equation 3.20 are fulfilled. A task allocation decision to choose one of these operations is made in the next step. Among other aspects, this decision needs to reflect uncertainty of data in the world model (Chapter 4). The robot can finally take steps to coordinate execution of the chosen operation with the human according to a teamwork mental model (Chapter 5), e.g. by passing it to the skill execution engine, by actively (re-)evaluating pre- or postconditions or by communicating missing information. Sensor data acquired during this step results in changes to P_t . This process is repeated, until all operations are DONE.

3.4. Conclusions

Summary

Establishing a shared mental model about the task to complete is of central importance for flexible human-robot teaming. This chapter has shown how prior art on task-level robot instruction with skills and on graphical robot programming can be applied to this problem. The end-user is supplied with a graphical user interface for instantiating symbolic, parametric skills and grouping them into precedence graphs (Section 3.2.1). The focus lies on object-related parameters that can be specified within a virtual representation of the workspace. Skills are internally represented by graphs of reusable basic actions (Section 3.1.1). This renders the top-level task model independent of specific hardware as different components can be controlled by supplying a skill execution engine

with respective action implementations. Skill graphs are designed to possess a control and data flow that ensures beneficial properties for graphical modelling as well as human-robot coordination (Section 3.1.2). This special structure enables automatic adaption of the task editor to new skills as well as deduction of operation pre- and postconditions (Section 3.2.2). Skill execution moreover terminates in clearly defined success or error states to facilitate integration with symbolic reasoning components. An example domain definition in terms of parts, actions and skills demonstrates practical applicability of the framework (Section 3.1.3). The domain targets scalable tasks for benchmarking cooperative work. Although manipulating abstract parts, skills cover relevant variants of pick-and-place operations, basic part processing steps, tool use and support in assembly with direct human-robot collaboration and communication. They thus foster composition of tasks where overall task duration, placement of parts in the workspace, duration of individual operation and the degree of close interaction can be adjusted arbitrarily.

Discussion

The rather open definition of basic actions supports convenient task-level parametrisation. It furthermore provides an interface to various algorithms for advanced robot capabilities. For instance, grasp or motion planning can be used when implementing intelligent actions with rich semantics. Compared to the traditional notion of action primitives with a direct mapping to control strategies, this comes, however, at the cost of increased design and implementation effort. In particular, the benchmark domain shows a fragmentation into numerous actions with different task-level parameters even within this bounded use-case.

The mechanism to provide pre- and postconditions guarantees prediction of expected effects on any input objects – in combination with using a camera as the only sensor to detect task progress, this limits the types of skills that can be modelled to operations with visually observable outcomes. E.g., visual inspection will not necessarily result in a change of part states. Such process steps may only occur in junction with an observable change within the same skill (cf. the `PickShake&Place` skill of the benchmark domain in which the unobservable process step of shaking is followed by a part transfer). Only then can postconditions serve their purpose of rendering task progress observable (Section 3.3).

When humans and robots equally possess the authority to choose operations from the task model, they may try to do the same operation simultaneously – this can generally result in competition for resource allocation. Moreover, grasping a part means claiming exclusive access to a resource. From the point of view of computer science, a potential for deadlocks is thus raised by the possibility of mutual waiting for part availability. Yet, skill graph semantics implies immediate abortion and reverse execution if resources are unavailable. Preemption is thus enforced by the skill execution engine in such situations to avoid ‘hold and wait’-situations. The fact that only one part is claimed by the executing agent per skill in the benchmark domain furthermore prevents circular waiting. The deadlock conditions according to Coffman [28] can thus never be satisfied. In summary, resource allocation during skill execution can therefore generally be assumed free of deadlocks – this is an important property for the design of coordination algorithms that are intended to keep the system capable of reasoning and acting anytime (Chapter 5).

With regard to collaborative operations, where agents have individual roles in close synchronous interaction, the framework currently limits skill description to one fixed role of the robot (e.g. ‘holding’, while a human agent is assumed to perform the dexterous assembly step). Depending on their capabilities, roles within an operation might however also be filled by two robots in the team during absence of humans – the idea of *skills with roles* in which a skill may contain several graphs for individual roles sets a possible direction for future work on even more flexible task load sharing (Section 7.2).

Human-Aware World Modelling for Task Allocation

4.1. World Model Definition and Maintenance	52
4.2. World Model Ageing	56
4.2.1. Human Workspace Model	57
4.2.2. Interaction Indicators	58
4.2.3. Trustworthiness of Data	61
4.3. Metrics for Task Allocation	63
4.4. Conclusions	64

WORLD MODELS are an essential component of cognitive robot systems, as they supply machines with an understanding of their surroundings. In principle, two classes of world modelling can be distinguished: *Sub-symbolic* world models applied in robotics often hold geometric data, e.g. point clouds, triangulated surfaces or occupied voxels representing the union of all objects within the workspace. In contrast, *symbolic* world models feature a higher level of abstraction. To this end, geometric representations are split up into the physical entities they contain. These entities are then represented by their state in terms of mostly human readable properties, e.g. the type, colour or location of some object. We have already seen that symbolic descriptions of objects have an important function in task-level skill parametrisation (Section 3.1.1). In this work, the world model is mainly used for tracking progress and making decisions regarding tasks composed of these skills. It is thus reasonable to adopt the symbolic paradigm – this way the pre- and postconditions defined for skills establish a direct link between task modelling and perception results stored in the world model. A concrete conception and incremental update procedure for integrating incomplete information from eye-in-hand cameras into a symbolic world model is outlined in Section 4.1. As the robot moves, parts will eventually get out of sight. Being the major basis of decision-making, the world model must therefore reflect potential human interaction with parts under this sort of partial observability. Inspired by the process of human forgetting, a data ageing procedure is applied to stored objects that are not sensed at some given point in time (Section 4.2): Based on a human model, this procedure assigns a *certainty indicator* as

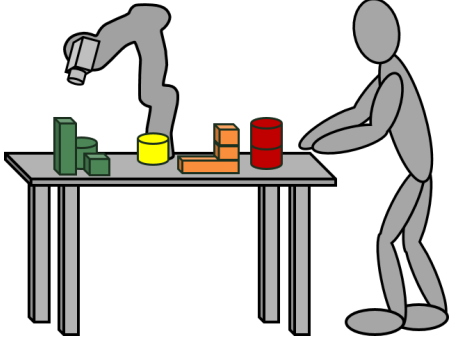


Figure 4.1.: The human-aware world model assigns a certainty value to objects that are currently out of view for the robot. Data on objects with high certainty (green) is likely still valid. By contrast, low certainty (red) indicates likely human manipulation of parts since their last observation.

visualised in Figure 4.1 to each object to render the world model human-aware. The indicator is based on the accessibility of parts for humans, and on their current relevance to the task. Certainty can be used to define metrics to support task allocation decisions (Section 4.3) – equipped with these metrics, the robot can avoid attempts to manipulate parts that may likely have been modified by humans since their last observation. The system can instead prefer working with objects that are not conveniently reachable for the human during the online coordination process (Chapter 5). The below content extends the author’s prior work on human-aware world models [136] and their application to human-robot teaming [134].

4.1. World Model Definition and Maintenance

Robot knowledge about the physical world at time t is encoded by a set of object instances (so called *entities*) $E_t = \{e_1, e_2, \dots, e_{|E_t|}\}$. Each entity is described by an object state $e \in \bar{D}$ according to Section 3.1.1. This world model is used to store perception results that the robot system has acquired about the physical world on the one hand. On the other hand, it also encodes the goal state of the task. To this end, E_t is divided into two distinct subsets: The *existing* entities (e-entities) E_t^e have previously been observed with the camera system – they can be used to verify operation preconditions and decide whether all needed resources are available for execution. By contrast, *wanted* entities (w-entities) E_t^w describe object states that will emerge in the course of task execution after success of some operation. The set E_t^w can be seen as a structured free space representation in which individual instances correspond to operation postconditions. They can thus be used to guide perception towards detecting objects that indicate task progress.

Given a task model $(T, <_T)$, the world model is initialized at time $t = 0$ as follows: Under the assumption that all necessary parts are generally available when initiating a task, E_0^e should contain each object in its initial state before being processed by any operation. Thus, an entity e_τ matching a precondition of operation $\tau \in T$ is inserted into E_0^e , if and only if there is no τ' preceding τ with a postcondition matched by e_τ . In other words, only object states that do not result from executing any operation are inserted. In contrast to this, w-entities are not only added for the final task goal state, but also for any observable intermediate step. However, object states that match a postcondition as well

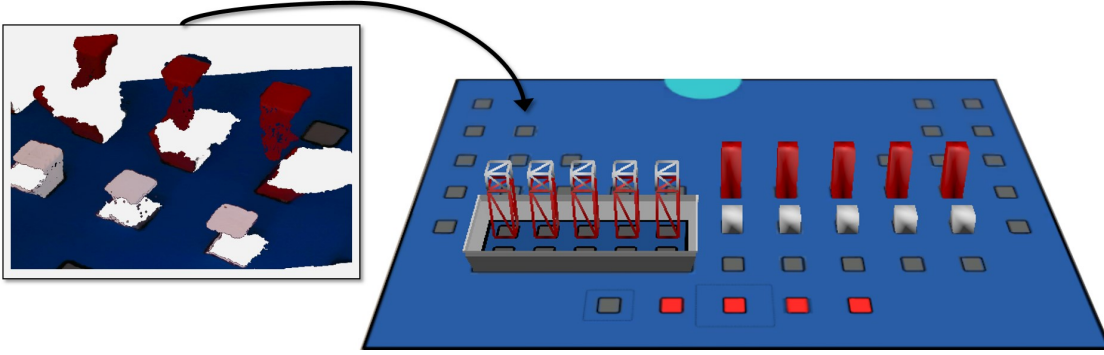


Figure 4.2.: The world model is initialised with the initial state of all parts (right, solid objects) and the goal state emerging from executing a shared task (right, wireframe objects). This representation is incrementally updated with symbolic information extracted from RGB-D point clouds that show a part of the workspace (left).

as a precondition of the same operation are excluded – such pairs of identical conditions indicate parts that cannot provide evidence for task progress. This situation occurs e.g. for the containers that are never moved in our example domain, or more generally for any tool that is returned to its initial position after usage during an operation. A possible result of the initialization procedure is shown to the right of Figure 4.2. In this Section, the focus lies on perception and thus in particular on consecutive changes to this initial model caused by availability of new sensor data. A need for modifications can also emerge from communication and reasoning during the coordination process. These effects are elaborated in Chapter 5 – the interface provided by the world model to other system components is briefly outlined at the end of this section.

Updates with perceptual data: Advancement from E_t to E_{t+1} is triggered whenever new sensor data is available. This means that a new point cloud showing a part of the scene is provided by the RGB-D eye-in-hand camera (Figure 4.2, left). An object recognition procedure is applied to this point cloud based on information on the object types that may occur in the domain (cf. Section 3.1.1). This procedure extracts the state of all visible objects at time t . The problem of recognising objects in point clouds has extensively been studied in literature (cf. the survey of Guo et al. [129]). It is thus assumed that any part that is sufficiently represented by points in the most recent point cloud is recognized robustly – errors during this procedure are not considered. The approach described hereinafter does, however, account for errors with the measurement of object poses in a world coordinate frame. This is particularly relevant in our case of a moving, low-budget eye-in-hand sensor system. Acquisition of the point cloud and the robot pose are here not synchronised in time precisely, as e.g. when using a hardware triggering mechanism – this impairs the inevitable pose estimation error from sensory noise by adding a component depending on robot movement speed.

Sensed objects always lie within the viewing frustum of the camera at time t . Let respective object states be represented by the set E_t^{vis} of currently sensed entities. Furthermore, let the predicate $\text{inFrustum}(e)$ be TRUE if the geometry of the object represented

by e lies in the frustum. With $\text{isOccluded}(e)$ returning `TRUE` if visibility of e is blocked by another part or limited by the camera sensing principle, E_{t+1} is constructed from E_t and E_t^{vis} based on the following interim steps: *Newly detected objects* E_t^{new} with

$$E_t^{\text{new}} = \{e \in E_t^{\text{vis}} \mid \nexists e' \in E_t^e : e \approx_{\bar{D}} e'\} \quad (4.1)$$

do not have a correspondence in E_t^e with respect to the precision encoded by the object state comparator $\approx_{\bar{D}}$ (Equation 3.2). They have not been sensed so far (e.g. parts at their goal positions after operations executed by humans) and must be added to the world model. By contrast, we refer to re-detected objects that are present in recent sensor data as well as in E_t^e as a set denominated the *confirmed objects* E_t^{conf} with

$$E_t^{\text{conf}} = \{e \in E_t^e \mid \exists e' \in E_t^{\text{vis}} : e \approx_{\bar{D}} e'\} \quad (4.2)$$

Any object in E_t^e that lies within the current viewing frustum should be confirmed according to Equation 4.2 unless it is covered by another object or has been moved in the meantime. Thus, *missing objects* E_t^{miss} stored in E_t^e that should be visible, but which are no longer observed in the physical world, are given by

$$E_t^{\text{miss}} = \{e \in E_t^e \setminus E_t^{\text{conf}} \mid \text{inFrustum}(e) \wedge \neg \text{isOccluded}(e)\}. \quad (4.3)$$

Missing objects represent data that has become invalid due to interaction of some other agent. They may thus be removed when transitioning to E_{t+1} . Confirmed data must be retained in the world model. However, re-detection of parts can be used to reduce object recognition measurement errors. To this end, a **filter** function takes as an input two similar object states ($e \approx_{\bar{D}} e'$). It returns a new, filtered state $e'' = \text{filter}(e, e')$, e.g. by averaging the locations of e and e' . Then, *updated objects* E_t^{update} can be calculated by applying the filter to confirmed entities, i.e.

$$E_t^{\text{update}} = \{\text{filter}(e, e') \mid e \in E_t^{\text{conf}} \wedge e' \in E_t^{\text{vis}} \wedge e \approx_{\bar{D}} e'\}. \quad (4.4)$$

Putting things together, the new set of existing entities E_{t+1}^e is composed as follows:

$$E_{t+1}^e = (E_t^e \setminus (E_t^{\text{miss}} \cup E_t^{\text{conf}})) \cup E_t^{\text{update}} \cup E_t^{\text{new}} \quad (4.5)$$

Missing entities as well as confirmed ones are removed. The latter are replaced by updated, filtered object states. Finally, newly detected parts are added. Sensor data cannot provide information about objects out of the viewing frustum – they are thus transferred into E_{t+1}^e unalteredly.

One step of this maintenance procedure is shown in Figure 4.3. Compared to the initial world model E_t^e (left), the physical world was changed by moving the part labelled seven onto the stack to the left. The new location of this entity cannot be observed from the current camera pose. This change is therefore not reflected by E_{t+1}^e . By contrast, it can be observed that a part matching state seven is missing. As there is no other object recognition result occluding this entity, it should be visible. It is thus moved into E_t^{miss} and deleted from the world model during the update step. The newly added part nine

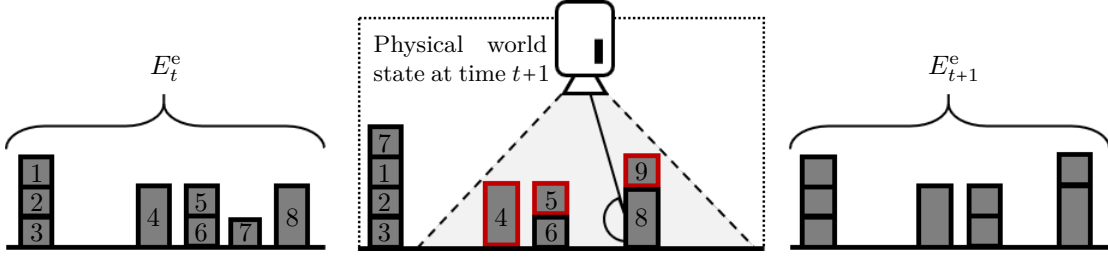


Figure 4.3.: During one step of world model maintenance the set of known objects (left, dark grey) is updated based on objects detected in the physical world (centre, red) and the location of parts with respect to the camera viewing frustum (centre, dashed lines).

is among the remaining entities within the viewing frustum at time $t + 1$. There is no correspondence for respective object state in E_t^e – a new entity is thus added to E_{t+1}^e . The remaining entries four and five of the recognition results set E_t^{vis} confirm previously known pieces of information. They are inserted into E_{t+1}^e after applying the filtering mechanism. Although theoretically hit by rays from the camera, entities six and eight are not included in E_t^{vis} . Due to the obtuse angle of rays against part side faces, these faces are commonly undersampled and insufficient for classification. It is important to notice that the `isOccluded` predicate is designed to model any kind of restriction in perceiving parts. This includes occlusion by physical objects as well as already mentioned limitations caused by the camera sensing principle. Objects six and eight are thus occluded by those denominated five and nine. They belong to neither of the categories defined by Equations 4.1 through 4.3 and are thus transferred into E_{t+1}^e , together with any object states outside the viewing frustum. Naturally, maintenance of e-entities in the world model depends on the ability to recognize parts, determine occlusions and check the location of objects with respect to the viewing frustum. Details on a concrete realization are given in the context of experimental evaluation (Section 6.1.1).

The update procedure for w-entities on incoming object recognition results is less complex: A w-entity e^w makes free space reserved for a part that emerges during task execution explicit. Thus, it is eventually replaced by an identical e-entity. As soon as this e-entity is observed, e^w is no longer needed and can thus be deleted. More formally, this update step is defined as

$$E_{t+1}^w = E_t^w \setminus \{e \in E_t^w \mid \exists e' \in E_t^{\text{vis}} : e \approx_{\bar{D}} e'\}. \quad (4.6)$$

Updates issued by other system components: Reasoning components can generally report any e- or w-entity to the world model. These entities will be inserted unless an equivalent object state according to the comparator $\approx_{\bar{D}}$ is already stored in E_t . Such manipulations are usually necessary, when the reasoning component has deduced completion of certain operations without having observed satisfied postconditions explicitly or when the system itself has changed object states by executing an operation. In particular, success or failure of some operation $\tau \in T$ can directly be reported to the world model. For operations carried out successfully, the world model will first delete all w-entities matching the preconditions. Then, object states satisfying the postconditions

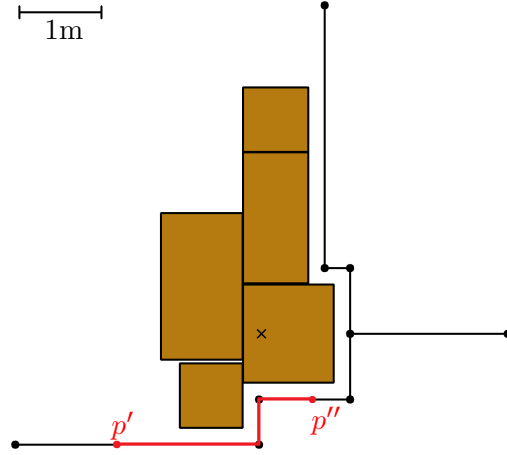
are inserted into E_t^e to reproduce operation effects in the world model. Thereby any overlapping w-entities are discarded in an analogous way to the policy during the update with perceptual data described above. In case of failed operations, entities are removed based on the skill type and error that was reported by the skill execution engine. E.g., if grasping a part failed, this object must have been moved by another agent since the last observation. Any object state representing that part is thus wrong and can be deleted.

4.2. World Model Ageing

We have seen so far, how object recognition results from images showing only a part of the workspace can incrementally be integrated into an overall, symbolic world model. But humans may change the environment in areas that are currently out of sight for the robot camera. The system must be aware of these potential changes – otherwise decision-making might frequently result in executing operations that fail, as some other agent has claimed necessary resources in the meantime. This issue of partial observability can be addressed by rendering the world model *human-aware*, i.e. by making it reflect potential human influences on parts. The approach taken in this thesis has initially been inspired by models of human memory, particularly by the early *decay theory* of Brown [22] that tries to explain forgetting over the short-term. The main hypothesis of Brown’s work is that perceiving something leaves a neurochemical trace in one’s brain. This *memory trace* disintegrates gradually, which leads to forgetting unless knowledge is rehearsed. Although recently discussed controversially in the field of cognitive psychology [101], this theory provides a pragmatic approach for the use case of human-aware world modelling: Applying a similar decay to out-of-sight data can be used to deduce a measure of uncertainty about prior perception results – we will refer to this process as *world model ageing* or *data ageing*. When parts get back into sight, their trace (in our case given as object states) is refreshed, and the decay procedure is restarted.

The object detection likelihood used by Baraglia et al. [9], implemented by a counter variable per object (cf. Section 2.1), can be seen as a realization in line with this idea. However, here the decay is applied to all known entities alike. This does not account for scenarios in which parts of the workspace are currently not reachable for humans due to ergonomic constraints. No change can happen in these areas unless caused by the system itself. Discarding objects would thus be an avoidable loss of data – unnecessary actions to later-on explore this area again would follow. This limitation can be overcome by adding a model to estimate the human handling area. Such human models have so far been used for incorporating ergonomic considerations into task allocation (e.g. [78, 97]). Similarly, the Mightability Maps of Pandey et al. [95] provide a compact encoding of reachable regions for robots as well as humans in different postures for use in action planning. Still, even when excluding certain entities from ageing based on a human model, applying a uniform decay to the remaining pieces of information does not cover important aspects of temporal dynamics: An object that is quickly bypassed by some agent is e.g. less likely to be modified than parts in a region in which a human has been staying for a longer period of time. Moreover, the relevance of parts for the

Figure 4.4.: The human workspace model encodes the assumption that human motion happens on a roadmap on the shop floor ground plane (black), close to a set of workbenches (brown) defined in 2D via their top-view dimensions. Workers are assumed to move on paths along the edges (red) between arbitrary points on the roadmap.



task may change over time, thus making their manipulation more or less likely. Based on these considerations, interaction indicators to quantify the likelihood of parts being modified by humans are introduced in Section 4.2.2. Calculation of these indicators is based on a human workspace model (Section 4.2.1), on sensor data on human presence and on observed prior human participation in the task. Finally, an extensible formal framework to integrate several indicators into one certainty value of world model entries is introduced in Section 4.2.3.

4.2.1. Human Workspace Model

The workspace model needs to reflect the idea of spatial flexibility which enables workers to change over into the coexistence mode. To account for this, the model covers a larger shop floor area rather than only the direct, close environment of the hybrid workstation. Human motion on the shop floor is therefore modelled as follows: A set of points $R^P = \{p_1, \dots, p_{|R^P|}\} \subset \mathbb{R}^2$ and the symmetric adjacency matrix $R^e = r_{ij} \in \{0, 1\}^{|R^P| \times |R^P|}$ define a roadmap (R^P, R^e) . Two points p_i and p_j are connected by a bidirectional edge if and only if the matrix entry r_{ij} is 1. Workers can move linearly along these edges to follow paths between any two points p', p'' on the roadmap. Most edges run parallel to the boundaries of workbenches on the shop floor – these edges cover positions which task work is carried out from. The remaining points define off-workbench locations. Respective sites target situations that make a worker leave the shop floor. Figure 4.4 shows a shop floor plan example – a suitable roadmap that matches the setting of a concrete application is assumed to be provided by system integrators together with the domain definition of parts and basic actions (Section 3.1.1).

Objects within a limited range on the workbenches can be reached from each position on the roadmap. This *handling area* is primarily determined by human body measures, particularly by the distance of the hip from the ground h_{leg} , the torso length h_{torso} , and the distance between shoulder and grip axis l_{arm} . These values can either be obtained from standards (e.g. ISO/TR 7250-2 [55]) or by measuring individual workers' proportions. From an ergonomic point of view, a strongly forward inclined posture of the upper body

is stressful and should thus be avoided at standing workplaces [31]. Therefore, the model takes an angle β_{\max} of maximum forward inclination into account. Then, the handling area can be approximated based on the limiting cases shown in Figure 4.5. Parts within arm reach are closer to the body axis between *hip point* H and *shoulder point* S_1 than the maximum grasp point G_1 . Objects in this range can be grasped conveniently while standing upright (left). When increasing forward inclination of the back, the shoulder point shifts until reaching S_2 at the maximum angle β_{\max} (right). The ultimate grasp point that can just be reached from an ergonomically safe posture is denominated G_2 . With the aforementioned body measures, S_1 and S_2 are given as

$$S_1 = \begin{pmatrix} 0 \\ h_{\text{leg}} + h_{\text{torso}} \end{pmatrix} \quad S_2 = \begin{pmatrix} \sin \beta_{\max} \cdot h_{\text{torso}} \\ \cos \beta_{\max} \cdot h_{\text{torso}} + h_{\text{leg}} \end{pmatrix}. \quad (4.7)$$

Both grasp points G_1 and G_2 can then be calculated by intersecting circles of radius l_{arm} around the shoulder point locations $S \in \{S_1, S_2\}$ with a line \vec{x} that describes the workbench surface. With given workbench height h_{table} , the parametrised line \vec{x} and these shoulder point circles c are defined by

$$\vec{x} = \begin{pmatrix} 0 \\ h_{\text{table}} \end{pmatrix} + \lambda \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad c : \|x - S\|^2 = l_{\text{arm}}^2. \quad (4.8)$$

Workbenches are assumed to have a reasonable height according to the recommendations of Daub et al. [31]. For $h_{\text{table}} \geq h_{\text{leg}}$, substituting \vec{x} into c and solving for λ leads to the line parameters of two intersection points in both situations of Figure 4.5. In our geometric situation G_1 and G_2 are each characterised by the positive of these solutions λ_S . This solution is

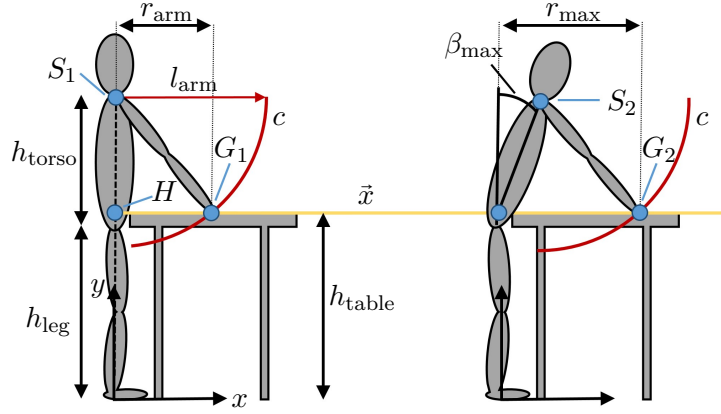
$$\lambda_S = s_x + \sqrt{s_x^2 - (|\vec{\Delta}|^2 - l_{\text{arm}}^2)}, \quad (4.9)$$

with the x-component s_x of $S \in \{S_1, S_2\}$ and $\vec{\Delta} = H - S$ [108, p. 248]. The above calculations result in two measures that are particularly important for rating the likelihood of parts being modified (Section 4.2.2): Parts within a range of $r_{\text{arm}} = \lambda_{S_1}$ from the body axis are highly accessible. By contrast, any object with a distance greater than $r_{\text{max}} = \lambda_{S_2}$ should not be handled by workers at all.

4.2.2. Interaction Indicators

So called *interaction indicators* quantify aspects that influence the likelihood of human interaction with certain parts in the workspace. Modification of an entry $e \in E_t$ should e.g. be rated highly probable, if this object is relevant to the next step within the task, and if some human is already in grasping range. We can here identify two aspects that take influence based on different information: Relevance can be deduced from an estimate of task progress and accessibility requires knowledge on the position of humans in the workspace. By separating these influences into individual indicators, an extensible framework can be built that rates overall trustworthiness by considering only those aspects that can be served by the available sensing and reasoning capabilities. To this

Figure 4.5.: The handling area is defined by two grasp points (G_1, G_2) in an upright posture (left) and when leaning forward with an incline of β_{\max} (right). Given the shoulder points S_1 and S_2 , grasp points are given as the intersection of circles around shoulder points (red) with the line \tilde{x} along the workbench surface (yellow).



end, each interaction indicator is generally described by a function $f^{\mathcal{HI}} : E_t \times \dots \rightarrow [0, 1]$. These functions must at least take an entity from the current world model E_t , i.e. the state of an object, as an input. They may optionally rely on additional data as e.g. the roadmap, results of human tracking etc. An output value $f^{\mathcal{HI}}(e, \dots) = 1$ means that interaction with an object is very likely regarding this indicator. Analogously, $f^{\mathcal{HI}}(e, \dots) = 0$ indicates low likelihood. We will further dwell on the examples of accessibility and relevance. Possibilities to capture these aspects numerically are shown hereinafter.

The notion of accessibility is mainly based on the 2D shop floor plan and the human handling area as defined in Section 4.2.1. By construction of the human model, points within a range of r_{arm} can easily be accessed from an ergonomically comfortable, up-right posture (Figure 4.5). These points are assigned the maximum indicator value. Distances in the interval $[r_{\text{arm}}, r_{\text{max}}]$ require an increasing inclination of the upper body. This leads to rising ergonomic strain (cf. e.g., respective ratings of trunk movement in the REBA procedure [50]). Assuming proper instruction of workers to avoid work-related health issues, the likelihood of parts being accessed thus decreases with growing distance. Any point farther away than r_{max} results in critical strain – therefore parts in this area are unlikely to be accessed at all, and thus rated with 0. For a point $p_e \in \mathbb{R}^2$ demarcating the projection of the location of some entity e onto the shop floor ground plane, and for some location $p_w \in \mathbb{R}^2$ on the roadmap (Figure 4.4), accessibility $\mathcal{A} : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow [0, 1]$ is approximated heuristically as

$$\mathcal{A}(p_e, p_w) = \begin{cases} 1 & \text{if } d(p_e, p_w) \leq r_{\text{arm}}, \\ 1 - \frac{1}{r_{\text{max}} - r_{\text{arm}}} \cdot (d(p_e, p_w) - r_{\text{arm}}) & \text{if } r_{\text{arm}} < d(p_e, p_w) \leq r_{\text{max}}, \\ 0 & \text{else} \end{cases} \quad (4.10)$$

Here, $d(x, y)$ denotes the euclidean distance between points x and y . The decrease in accessibility for distances between r_{arm} and r_{max} is assumed to be linear. The function profile is shown qualitatively in Figure 4.6. Of course, a more complex ergonomic model could be taken as a basis, if found necessary.

With this point-based accessibility definition, the first interaction indicator $f_{\mathcal{A}}^{\mathcal{HI}}$ can be formulated. The world model provides information on the type and pose of each object

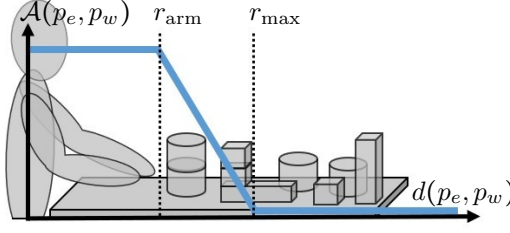


Figure 4.6.: Point-based accessibility (blue) is maximal within arm reach and decreases with growing distance to the human body axis.

$e \in E_t$ in the world coordinate frame. Assuming that a geometric model for each part type is given, the 2D coordinates of the part centroid $\text{centroid}(e)$ in the workbench plane can be calculated. This way, parts are reduced to a location that must be reached when trying to manipulate them. The point p_{reach} on any of the roadmap edges that minimizes $d(p_{\text{reach}}, \text{centroid}(e))$ is denoted $\text{reach_point}(e)$. Then, the interaction indicator is defined via point-based accessibility as

$$f_{\mathcal{A}}^{\mathcal{HI}}(e) = \mathcal{A}(\text{centroid}(e), \text{reach_point}(e)). \quad (4.11)$$

Figure 4.7 (left) illustrates the semantics of $f_{\mathcal{A}}^{\mathcal{HI}}$. Accessibility $\mathcal{A}(p_e, p_w)$ spreads a scalar field across the shop floor, starting from the roadmap. Parts are less likely to be manipulated with increasing distance, encoded by lower scalar field values.

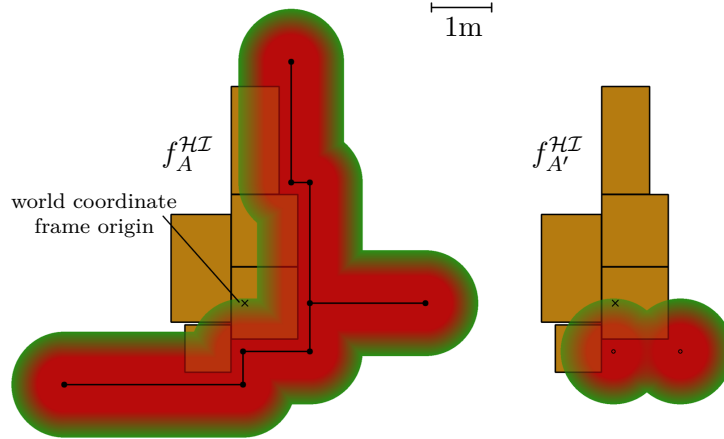
The interaction indicator $f_{\mathcal{A}}^{\mathcal{HI}}$ solely relies on the workspace model. This can be seen as a worst-case approximation, as each part that can generally be reached from the roadmap is assumed accessible. The approach clearly overestimates human presence, but therefore goes without additional sensor data. A more precise accessibility indicator can be calculated if the actual positions of workers on the shop floor are known. Let $H_t = \{h_1, h_2, \dots, h_{|H_t|}\} \subset \mathbb{R}^2$ denote the *human presence map* at time t . Each entry $h \in H_t$ marks the 2D position of a worker on the shop floor ground plane. A concrete method to acquire H_t from LIDAR scans is outlined in Section 6.1.1. With this additional knowledge the alternative accessibility indicator $f_{\mathcal{A}'}^{\mathcal{HI}}$ is calculated according to

$$f_{\mathcal{A}'}^{\mathcal{HI}}(e, H_t) = 1 - \prod_{i=1}^{|H_t|} (1 - \mathcal{A}(\text{centroid}(e), h_i)). \quad (4.12)$$

This formula applies the rules for joint probabilities and complementary events from probability theory – the expression thus models the probability that e will be accessed by any human nearby under the simplifying assumption that individual workers act independently. Similar to $f_{\mathcal{A}}^{\mathcal{HI}}$, $f_{\mathcal{A}'}^{\mathcal{HI}}$ can be visualized as a scalar field as shown in Figure 4.7 (right). The effect is limited to the area surrounding humans based on H_t .

The last interaction indicator addressed in this work tries to capture current relevance of parts to the task. Relevance quantifies heuristically whether it makes sense to manipulate some entity from a cooperative worker’s point of view at the present time. With the goal of advancing the task, parts needed for operations that are up next will more likely be modified than those needed later-on in the task. Calculation of the relevance indicator is therefore based on the task progress estimate P_t maintained by the robot as introduced in Section 3.3. Furthermore, the definition needs a way to associate parts in the world

Figure 4.7.: Accessibility defines a scalar field across shop floor and workbenches (brown). Areas with high values (red) can be accessed easily, whereas parts with low values (green) are unlikely to be manipulated. Depending on the available sensor data, the field spreads around the roadmap (left) or around actually measured human positions (right).



model with operations they are needed for. This relationship is encoded by the function $\text{modifies}(\tau, e)$ which inspects all basic actions that the skill behind $\tau \in T$ of some task $(T, <_T)$ consists of. If there is any action with an input parameter matching the object state e , but with no identical output parameter, then $\text{modifies}(\tau, e) = \text{TRUE}$. Otherwise, e is not manipulated in the course of executing τ , resulting in $\text{modifies}(\tau, e) = \text{FALSE}$. A quantitative measure for relevance is then given as

$$f_R^{\mathcal{HI}}(e, P_t) = \begin{cases} 0 & \text{if } \nexists \tau \in T : P_t(\tau) \neq \text{DONE} \wedge \text{modifies}(\tau, e), \\ 1 & \text{if } \exists \tau \in T : P_t(\tau) = \text{ACTIVE} \wedge \text{modifies}(\tau, e) \quad . \\ 0.5 & \text{else} \end{cases} \quad (4.13)$$

If there is no more operation to do that requires a part represented by object state e , then there is no need to assume interaction with e ($f_R^{\mathcal{HI}}(e, P_t) = 0$). Those parts are not processed further by future operations (e.g. objects at their target positions in pick-and-place tasks). On the other hand, any part needed for operations that are ACTIVE is assigned the maximum relevance score of 1 – respective objects are needed for the next steps within the task. A medium value is assigned to all other objects in E_t .

4.2.3. Trustworthiness of Data

The goal of human-aware world modelling is to provide robots with a measure for judging trustworthiness of previously stored objects in the world model when they are out of sight. With the interaction indicators at hand, we can now assemble this measure. Therefore, a real-valued certainty indicator $\mathcal{C}_t : e \rightarrow [0, 1]$ is incrementally calculated for each entity $e \in E_t$ at time t . Following the idea of memory decay, $\mathcal{C}_t(e)$ decreases monotonously over time. Interaction indicators are designed to produce high values for data that may soon become invalid. By contrast, low values mean that there is no indication for human interaction and thus no need to forget information on respective parts. Consequently, the $f^{\mathcal{HI}}$ functions can be seen as the change rate or time derivative of \mathcal{C}_t . Integrating them over time leads to a human-aware world model with the following semantics: Information carried by entities with high certainty is probably still valid and reusable. On the other

hand, entities with low C_t values indicate parts that may have been changed and need to be newly observed. Certainty is reset to 1 if information is rehearsed, i.e. if e is confirmed by the camera image at time t ($e \in E_t^{\text{conf}}$). This procedure is applied to all stored objects in E_t alike. The set E_t^{conf} does, however, not cover w-entities, as they represent the intended goal state and cannot be sensed directly. Some $e^w \in E_t^w$ is therefore defined to be confirmed if e^w lies in the viewing frustum ($\text{inFrustum}(e)$) and if it is not replaced or occluded by any element of E_t^{vis} . These parts are confirmed to be still missing from their target positions. Hence we can define E_t^{free} as

$$E_t^{\text{free}} = \{e \in E_t^w \mid \text{inFrustum}(e) \wedge \neg \text{isOccluded}(e) \wedge \nexists e' \in E_t^{\text{vis}} : e' \approx_D e\} \quad (4.14)$$

in analogy to Equation 4.3. Based on these preliminary considerations, the calculation rule for C_t can now be fully specified:

Let $F^{\mathcal{HI}} = \{f_1^{\mathcal{HI}}, f_2^{\mathcal{HI}}, \dots, f_N^{\mathcal{HI}}\}$ denote the set of N interaction indicators to be taken into account. Then, the following expression enables entity ageing in a modular way with different interaction indicator combinations, depending on the available sensor data:

$$C_{t+1}(e) = \begin{cases} 1 & \text{if } e \in E_t^{\text{conf}} \vee e \in E_t^{\text{free}}, \\ \max(0, C_t(e) - \lambda \cdot \mathcal{HI}(e, F^{\mathcal{HI}})) & \text{else} \end{cases} \quad (4.15)$$

\mathcal{HI} is a reduce operation that merges several interaction indicators for e into one *human influence term*. To this end, $f_1^{\mathcal{HI}}, f_2^{\mathcal{HI}}, \dots, f_N^{\mathcal{HI}}$ are first individually applied to e . The resulting component indicators are then accumulated by multiplying them, leading to $\mathcal{HI}(e, F^{\mathcal{HI}}) \in [0, 1]$. The influence term scales a constant decrement λ called the *trust factor*. For $\lambda = 0$ the world model corresponds to a robot memory that never loses trust in data once gathered. Another limit case emerges for large values of λ – the larger λ , the smaller the threshold value $\mathcal{HI}^* = \frac{1}{\lambda}$ of $\mathcal{HI}(e, F^{\mathcal{HI}})$ that causes stored object to be discarded within one ageing step. This means in practice that the robot will forget any perception results when detecting even the slightest potential for human interaction.

Hereinafter, $F_1^{\mathcal{HI}} = \{f_A^{\mathcal{HI}}, f_R^{\mathcal{HI}}\}$ and $F_2^{\mathcal{HI}} = \{f_{A'}^{\mathcal{HI}}, f_R^{\mathcal{HI}}\}$ will be used to investigate the influence of human positional data on the coordination process. Both combinations have in common that $\mathcal{HI}(e, F^{\mathcal{HI}}) = 1$ causes C_t to drop by the maximum decrement for relevant, easy to reach objects. Ageing is deferred for entities that are currently either not relevant to the task or for those that can only be reached under increased ergonomic strain. The value of $f_A^{\mathcal{HI}}(e)$ only depends on the distance of e to the roadmap. It is thus constant over time. Scaling this value with the constants that $f_R^{\mathcal{HI}}$ produces leads to graphs of $C_t(e)$ that are piecewise linear when applying $F_1^{\mathcal{HI}}$ – the gradient changes whenever the relevance of e rises or falls. Certainty calculated based on $F_2^{\mathcal{HI}}$ exhibits more complex temporal dynamics, as $f_{A'}^{\mathcal{HI}}$ reflects human motion on the shop floor. Qualitative graphs for different scenarios of human motion are shown in Figure 4.8 for a constant value of $f_R^{\mathcal{HI}}$. In Scenario 1, a worker quickly bypasses an object e_1 on the workbench, stays out of reach for some time and then returns on the same path. Certainty of e_1 drops during the way forth and back. While the human is out of grasping range, $f_{A'}^{\mathcal{HI}}(e, H_t) = 0$ leads to an overall human influence term of 0. Certainty hence stays constant during this time span. Scenarios 2 and 3 assume identical motion towards

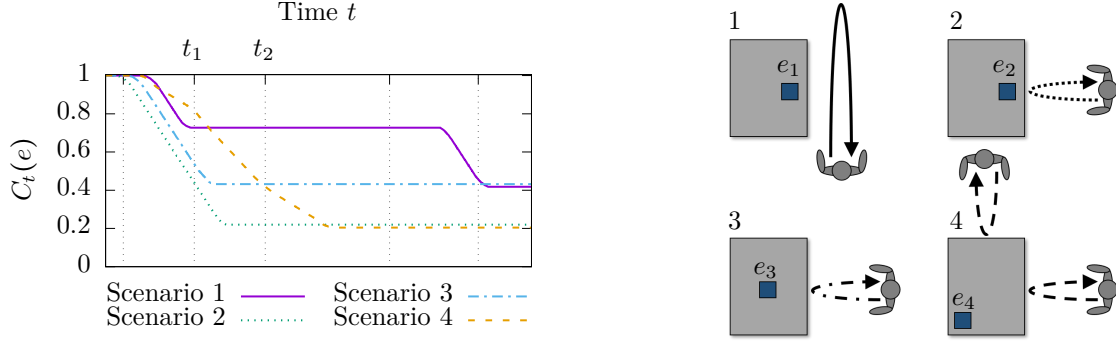


Figure 4.8.: Certainty $C_t(e)$ is a monotonously falling function. High values mean high trustworthiness of data regarding parts ($e_1 - e_4$) and low values mean low trustworthiness. The change rate of C_t depends on the likelihood of human interaction with parts. Different scenarios 1 - 4 of human presence (right) lead to characteristic, qualitative robot memory decay graphs (left).

the workbench but consider parts e_2 and e_3 in different positions. In this setup, e_3 is harder to reach than e_2 – thus, e_2 remains within the human handling area for longer than e_3 . As a result, certainty of e_2 starts dropping earlier and reaches a lower absolute value (Scenario 2) when compared to $C_t(e_3)$ (Scenario 3). The distance between objects and the worker falls below r_{arm} in Scenarios 1 to 3. This manifests in certainty reaching a constant, identical gradient as soon as $f_{A'}^{\mathcal{H}^I}(e, H_t) = 1$. Scenario 4 illustrates a different situation: Two workers arrive at the workbench and stay for a while before starting their way back. The part represented by e_4 does not lie within a distance below r_{arm} for either of them – therefore the maximum certainty change rate is never reached. One worker arrives late at time t_1 and leaves early at t_2 . The certainty falloff during these points in time is stronger compared to the presence of only one person (cf. Equation 4.12).

4.3. Metrics for Task Allocation

The human-aware world model as defined in the preceding sections serves as a basis for task allocation decisions. Parts with low certainty scores might have been manipulated since their last observation. The robot should thus check their actual availability before trying to manipulate them itself to avoid skill execution failures. By contrast, manipulation of parts with high certainty will likely succeed. Recall that preconditions of operations are logics formulas demanding existence of object states in the world model matching skill input parameters (Section 3.2.2). Let $E_\tau^e \subseteq E_t^e$ denote the set of current e-entities matching the preconditions of some operation τ . The parts represented by E_τ^e are required to carry out τ – this operation is thus a promising candidate for successful robot execution if the entities $e \in E_\tau^e$ possess high certainty values. The notion of *readiness* $\mathcal{R} : T \rightarrow [0, 1]$ condenses certainty of several parts into one rating for operations. By calculating readiness as

$$\mathcal{R}(\tau) = \prod_{e \in E_\tau^e} C_t(e), \quad (4.16)$$

values of \mathcal{R} can be used as decision metrics in the task allocation process: High values of $\mathcal{R}(\tau)$ confirm readiness of τ for execution. Low values, by contrast, are indicators to guide perception towards the corresponding workspace areas.

Aside from achieving progress by successful skill execution, it is also important for the robot to observe task advancement caused by other agents. To this end, perception operations are issued to move the camera around and check operation postconditions actively. Excessive camera motion without information gain reduces system productivity. As with skills from the task model, it is thus desirable to render perception operations successful. Task progress not yet achieved is encoded in the w-entities E_t^w . Analogous to E_τ^e , let $E_\tau^w \subseteq E_t^w$ denote a set of w-entities matching the postconditions of τ . Both types of entities are aged alike – an element of E_τ^w with high certainty therefore indicates that respective object is probably still not present. As w-entities are generated from postconditions, this also means that progress on the corresponding operation is not to be expected. Similar to readiness, the *success* measure $\mathcal{S} : T \rightarrow [0, 1]$ compresses several w-entities related to some operation $\tau \in T$ into one score for τ . Success \mathcal{S} is given as

$$\mathcal{S}(\tau) = 1 - \prod_{e \in E_\tau^w} \mathcal{C}_t(e). \quad (4.17)$$

With this expression, an operation scores a high success rating if prior interaction near the free space representation of the goal state is likely. It is thus beneficial to monitor postconditions of operations with high \mathcal{S} values when trying to unveil task progress.

4.4. Conclusions

Summary

Relying on a robot eye-in-hand camera raises a major challenge, as previously sensed and stored objects in a world model may be modified by humans while out of sight. Hence, Chapter 4 has introduced the notion of human-aware world modelling. A human-aware world model provides support in handling this kind of partial observability by (i) integrating object recognition results from partial views of the workspace into a unified, symbolic world representation (ii) and by applying an ageing strategy to stored objects, which results in a measure to judge trustworthiness of world model content.

The world model covers sensed, existing parts as well as a representation of objects yet to emerge in the course of task execution. Data ageing is applied to all these entities alike by attaching a real-valued certainty measure to each entry. Starting from a high value in the moment of sensing a part, certainty decreases over time in proportion to the likelihood of human interaction. This likelihood is captured by an extensible set of interaction indicators, e.g. human accessibility of parts under ergonomic constraints or current relevance of objects to the task. Each indicator handles parts of the available sensory information. As this thesis seeks to evaluate human-robot teaming with limited sensor data, only knowledge on human positions in the workspace and a task progress estimate maintained by the robot are used. Yet, the calculation rule reducing several

indicators to one certainty value is modular and not limited to these examples – the approach can conveniently be extended to scenarios with more sensory input.

With this world model at hand, robot decisions are supported in several ways: Certainty of sensed objects helps to judge whether some operation has good prospects of success or whether failure due to missing parts is likely. Due to the consideration of ergonomic aspects during data ageing, this also points the robot towards operations that should be avoided by humans to reduce health issues. Similarly, planning of active perception for progress monitoring can be based on certainty: If entities that represent an operation goal state have low certainty, this indicates potential interaction in the respective workspace area – it may thus be beneficial to observe this area when trying to detect task advancement caused by other agents.

Discussion

Although exposing important features for robot reasoning, the human-aware world model as outlined above has certain limitations: (i) The system relies on robust object recognition that detects each object correctly as soon as it occurs on the camera image. Handling of uncertain recognition results may be realised by inserting several hypotheses with initially decreased certainty values but is currently not considered. (ii) The human handling area model as used in the accessibility indicator targets grasping parts on a workbench as a major focus of this work lies on pick-and-place operations – ergonomic considerations are therefore limited to avoiding strong forward inclination of the upper body. This is sufficient to define human reach but may be extended towards a more complex human kinematics model and full posture assessment, e.g. using the REBA procedure [50]. (iii) Certainty is merely a heuristic measure of likelihood that cannot fully model human decision processes. Contrary to our assumptions, some agent may e.g. prefer to perform certain operations, even if they are ergonomically more stressful than other options. Thus, even information carried by entities with high certainty may be faulty. Accepting possibly wrong world model content is, however, a trade-off introduced by the idea of limited sensor use – more information, or even full workspace observability in the extreme case, results in a world model with less errors, but inevitably comes along with a more costly sensor setup.

Coordination of Flexible Human-Robot Teams

5.1. Team Mental Model	68
5.1.1. Agent Capability Model	70
5.1.2. Interaction Categories	72
5.1.3. Flexible Communication Patterns	74
5.1.4. Preemptive State Machines	77
5.2. System Architecture	80
5.3. Dynamic Task Sharing	82
5.3.1. Decision-Making Strategies	83
5.3.2. Task Advancement	85
5.3.3. Knowledge Update	86
5.4. Conclusions	89

COORDINATED action is essential for productive teaming. Particularly dynamic co-working without a fixed, optimized schedule can only enhance productivity, if partners act complementarily rather than obstructing each other. The previous chapters have introduced concepts for knowledge representations that support robot participation in this process. Graphical modelling of precedence graphs (Chapter 3) leads to a shared task mental model including pre- and postconditions for each operation. Skills underlying each operation enable robot action, and conditions establish a link to perception-based decisions via a human-aware world model (Chapter 4): Validating the state of conditions by checking existence of parts in the world model provides the robot with an estimate of task progress on the one hand. On the other hand, explicit human-awareness provides metrics for dealing with partial observability, and for guiding robots towards operations that are ergonomically unfavourable for humans at the same time. This chapter puts things together by describing how the proposed system uses these components to engage into coordinated teaming. An overview of the process is shown in (Figure 5.1). After a task allocation decision to choose an operation, the robot needs to issue actions suitable for advancing this operation towards completion. Depending on individual agent capabilities, task advancement may e.g. emerge from activities like executing the operation,

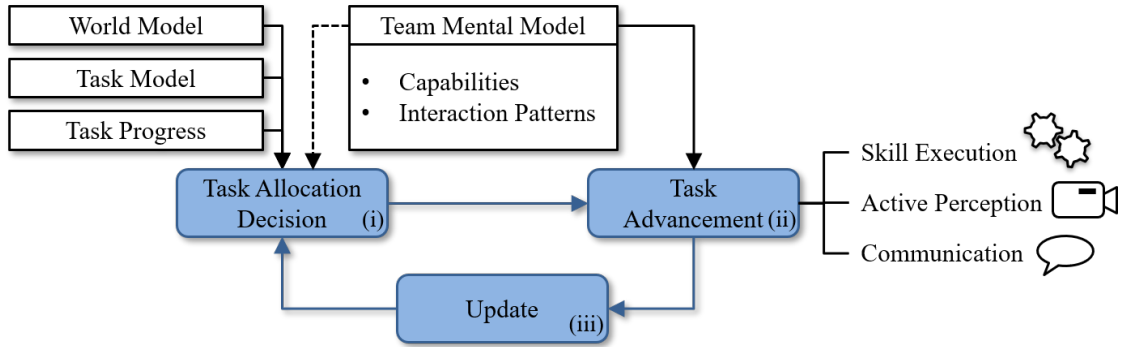


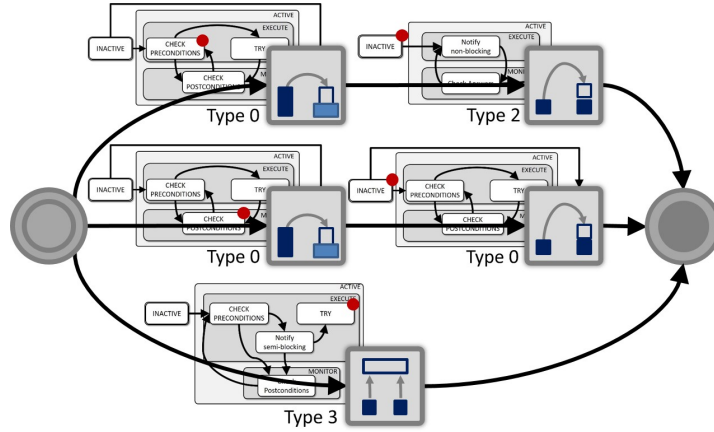
Figure 5.1.: Robot participation in coordinated teaming is achieved by repeating three steps: (i) A task allocation decision based on knowledge of the world state and estimated task progress results in the operation which is handled next. (ii) The team mental model influences the decision and defines a suitable strategy to advance the chosen operation towards completion. (iii) Action leads to new information to update knowledge before the next task allocation decision.

unveiling progress achieved by other agents using active perception or asking a partner to handle operations exceeding one’s own abilities. An extensible and exchangeable team mental model (Section 5.1) defines when to initiate which activity (skill execution, perception or communication) to advance some operation. Activities are therefore realised by different architectural components (Section 5.2). Execution of any activity will result in new information, either gained by perception while moving the robot hand or emerging from some other communication channel – these pieces of information are used to update the task progress estimate and world model before re-iterating the cycle. Details on the functioning of these three steps are provided in Section 5.3. The resulting system behaviour is finally discussed in Section 5.4.

5.1. Team Mental Model

Team mental models are used for explaining a human’s understanding of interaction within a team. According to Mathieu et al. [82], they cover typical interaction patterns, teammates’ capabilities and preferences, information flow through different communication channels etc. Reproducing all of the underlying human cognitive processes is challenging – several of the priorly used task models therefore try to account for these aspects at least partially, e.g. by explicitly focussing on preferences [40, 88] or communication [103, 114] (cf. Section 2.1). Most models show a tight coupling between the task and team mental model components (e.g. [60, 44, 88]). Creating task models may then e.g. also demand time-consuming training [44, 88] or even manual modelling [60] of interaction patterns, or explicit specification of agent capabilities [112] based on expert knowledge. This is not in line with the proposed task modelling approach (Chapter 3). Other dynamic teaming approaches solely rely on task progress monitoring and do not make human-robot communication a subject of discussion (e.g. [30, 73, 90]). Communication is, however, required to realise all levels of coordination (Section 1.2).

Figure 5.2.: The team mental model assigns a state machine to each operation of a task. Individual state machines prescribe the necessary course of action, perception and communication when working on respective operations depending on individual agent capabilities (Types 0 - 3). The current state (red dots) prescribes the next action needed to advance the completion of operations.



Drawing inspiration from the works of Roncone et al. [103] and Hawkins et al. [47], the concept proposed in this work relies on a *hybrid* task model. Each operation of a precedence graph is automatically embedded into a suitable team mental model. Task modelling is decoupled from specifying robot interaction behaviour this way. Component task models are formulated as deterministic state machines. In contrast to probabilistic models [103, 88, 114], this model can be modified conveniently and independently from concrete tasks to adjust robot behaviour without training.

To this end, operations are first grouped into *interaction categories* depending on agent capabilities. Interaction categories have previously been introduced in the author's prior work [135]. They are designed to cover operations that can be handled by single agents as well as such requiring an increasing level of coordination: e.g., an operation that is fully feasible for the robot can be handled by issuing a call to the Skill Execution Engine, and by triggering perception of pre- or postconditions in case of failure. By contrast, process steps with close, operation-level coordination (e.g. the **Assemble** skill introduced in Section 3.1.3) may require communication of mutual commitment with some partner before invoking the underlying skill. Otherwise, productivity might decrease due to agent idle times. Recall the fact that the task mental model according to Section 3.2 is purely process-oriented and thus defined independently of agent capabilities – the first component of our team mental model is therefore a method to determine feasibility of operations for individual agents (Section 5.1.1). The semantics and formalism behind interaction categories is introduced in Section 5.1.2. A deterministic state machine is defined for each interaction category (Section 5.1.4) as a component mental model. These state machines encode the protocol for interleaving action, perception and communication when trying to complete operations of some category. They add an additional layer to the hierarchy of actions, skills and precedence graphs (Figure 5.2): The current state within the state machine matching the assigned interaction category is stored for each operation. Having taken a task allocation decision in favour of some operation, the next action towards completing it is prescribed by this state. Actions yield observations, which in turn advance state machines into a subsequent state. For the sake of flexibility, component mental models are *preemptive*. The system can stop working on an operation

after defined state transitions. This way, the robot stays capable of acting by entering the next coordination loop iteration (Figure 5.1). This is important in situations where e.g. parts are missing for successful skill execution, or where human partners do not react to communication requests. In such situations, the system is designed to turn towards other operations, and re-address the process step in question later on. In particular, this also involves flexible communication patterns that avoid blocking dialogue and allow the system to recover, if the recipient of messages does not respond within a reasonable timespan (Section 5.1.3).

5.1.1. Agent Capability Model

For a given task $(T, <_T)$, feasibility of operations $\tau \in T$ for different agents is encoded by a *capability model*. We will use a binary model C_a that determines whether some operation τ is feasible for agent a , or not, i.e.

$$C_a : T \rightarrow \{\text{TRUE}, \text{FALSE}\}, \quad a \in \{\text{H}, \text{R}\}. \quad (5.1)$$

This model consists of only two functions C_H for humans and C_R for a robot. Yet, it covers the act of teaming up one robot with an arbitrary number of humans: The sensor data encoding human presence is restricted to the position of humans near the workspace only (Section 4.2.2) – there is especially no mechanism to distinguish between different workers. We can hence assume that workers with domain knowledge are similarly skilled based on the cooperative worker assumption (Section 1.2). Then, a single function C_H applies to any involved human.

A concrete capability model depends on the domain on the one hand, as some skills may by definition be infeasible for one single agent. This applies e.g. to the **Assemble** skill in our benchmark domain, and more generally to differences between human and robot abilities with respect to dexterous manipulation. On the other hand, feasibility can be influenced by the concrete hardware setup underlying the skill framework as well as by aspects of human physiology – a part that cannot be reached by a small manipulator may well be within the range of action for humans or a larger robot, a pick-and-place skill may be feasible for a human depending on part weight etc. Prior works on capability-based task allocation rely on agent models capturing these aspects (cf. Section 2.2). For instance, Makrini et al. [78] consider information on the maximum payload, range of actions or gripping force of agents and match them against the position, weight and dimensions of parts. Equation 5.1 provides the interface to integrate any similar considerations into the classification process that defines interaction categories later-on. This work, however, seeks to investigate teaming among equal peers, and the example domain comprises only lightweight, easy to manipulate parts. Participation in all skills presented in Table 3.2 is therefore assumed generally feasible for any agent when combined with any object type. The collaborative **Assemble** skill is infeasible for single agents by construction. For operations of this type or any similar skills, $C_H(\tau) = C_R(\tau) = \text{FALSE}$ holds independently of the concrete parametrisation – we assume that skills with this property are tagged accordingly during skill definition. Further considerations on feasibility are then not

needed for respective operations. For simplicity of notation, this distinction is omitted in the following formulas. They only apply to the remaining pick-and-place variants, where operation feasibility depends on input parameters and desired effects. This leaves us with the question of whether part locations in the workspace can be reached by individual agents when trying to determine operation feasibility.

With the object-centric skill definition of Section 3.1, these locations are given by skill input and output parameters. Let operation τ be an instance of skill s_τ . In analogy to the definition of pre- and postconditions, τ has object-related input parameters \bar{s}_τ^{in} and output parameters $\bar{s}_\tau^{\text{out}}$. The values of these parameters are object states taken from \bar{D} . They describe the objects needed to carry out the operation and the expected state of objects after successful execution. An agent must thus be capable of reaching the location of these parts in their initial (input) state as well as to reach their desired (output) position to be capable of the overall pick-and-place operation. Let $E^\tau \subset \bar{D}$ denote a set that gathers all aforementioned input and output object states related to τ . Reachability of parts in the workspace has already been discussed in Section 4.2.1 from the human point of view. Following this model, ergonomic considerations limit the handling area: Possible worker positions are limited to a roadmap around workbenches on the shop floor. From each position, the farthest reachable point has a distance of r_{max} from the human body axis. As defined in Section 4.2.2, $\text{reach_point}(e)$ denotes the point on the roadmap with the least distance to part e . To this end, the **centroid** function maps parts to the 2D position of their centroid on the workbench surface (Figure 5.3, grey). With the euclidean distance $d(\cdot, \cdot)$, the human capability model is then given as

$$C_H(\tau) = \begin{cases} \text{TRUE} & \text{if } \forall e \in E^\tau : d(\text{centroid}(e), \text{reach_point}(e)) \leq r_{\text{max}} \\ \text{FALSE} & \text{otherwise} \end{cases}. \quad (5.2)$$

This model says that an operation is feasible for any involved human if and only if all input parts and goal states lie within the previously defined human handling area.

The robot capability model is motivated similarly. Here, the ‘handling area’ is restricted by the kinematic parameters of the robot system. We will denote by $\text{ik_tractable}(e)$ a function that returns TRUE if and only if an inverse kinematics solution to approach e exists. This information is produced by the Skill Execution Engine that has the necessary information on the kinematic structure including the robot end effector. Analogous to Equation 5.2, the robot capability model for pick-and-place operations is given as

$$C_R(\tau) = \begin{cases} \text{TRUE} & \text{if } \forall e \in E^\tau : \text{ik_tractable}(e) \\ \text{FALSE} & \text{otherwise} \end{cases}. \quad (5.3)$$

Equations 5.2 and 5.3 model the range of actions for individual agents. The workspace is thus decomposed into regions that are either exclusively accessible to humans or the robot, or that are shared among agents. An example of the resulting workspace areas is depicted in Figure 5.3. Here, the robot is capable of accessing parts in a circular region around its base position. A fraction of this region, the *shared workspace*, is also accessible to humans. Locations that are too distant from the roadmap as well

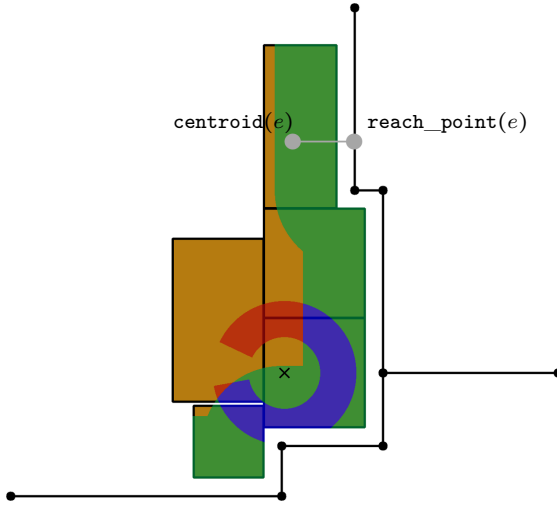


Figure 5.3.: The capability model is defined by the range of action of individual agents. Some regions W_H are exclusively accessible for humans (green). If points are too distant from the road map, only the robot system (\times) may be capable of reaching objects in respective regions W_R (red). Humans and robots share a part of the workspace W_{shared} that both can reach (blue). The remaining workbench surface areas are assumed inaccessible.

as those only accessible by human mobility are called *exclusive workspaces* of humans and the robot. We will refer to the shared workspace as W_{shared} . Respectively, the human-exclusive and robot-exclusive areas are denoted by W_H and W_R . These notions are crucial for the following introduction of interaction categories.

5.1.2. Interaction Categories

Interaction categories group operations into abstract classes. They represent different needs for interacting and communicating, depending on the ability of agents to handle operations. With this classification scheme at hand, each operation of a task can automatically be linked to a suitable protocol for interleaving perception, action, and communication. End-users can thus concentrate on pure process modelling, while suitable robot strategies for each interaction category are designed by experts (Section 5.1.4). Formally, one out of four *interaction types* is assigned to an operation $\tau \in T$ of some task model $(T, <_T)$, i.e.

$$\mathcal{I}_a : T \rightarrow \{\text{TYPE 0}, \text{TYPE 1}, \text{TYPE 2}, \text{TYPE 3}\} \quad (a \in \{H, R\}). \quad (5.4)$$

The classification function \mathcal{I}_a relies on the capability model to link operations to interaction types. Consequently, \mathcal{I}_a is agent-dependent ($a \in \{H, R\}$) and defines the following semantics of interaction categories for humans (H) as well as the robot (R):

TYPE 0: An operation is classified as TYPE 0 for agent a ($\mathcal{I}_a(\tau) = \text{TYPE 0}$) if and only if the agent can carry out τ *atomically*. Skill execution happens without interruption or help from other agents. The timeline in Figure 5.4 visualizes the chronological semantics of TYPE 0 from the robot point of view: Only the robot acts, while the other agent is not involved. In our benchmark domain, TYPE 0 is assigned to any operation that does only involve parts at locations within the reach of the agent in question, i.e. within the shared and robot-exclusive areas when considering \mathcal{I}_R .

TYPE 1: In the contrary case of TYPE 1 operations, agent a can *not* execute an operation τ all by his or her own – there is, however, another agent a' who can do so by

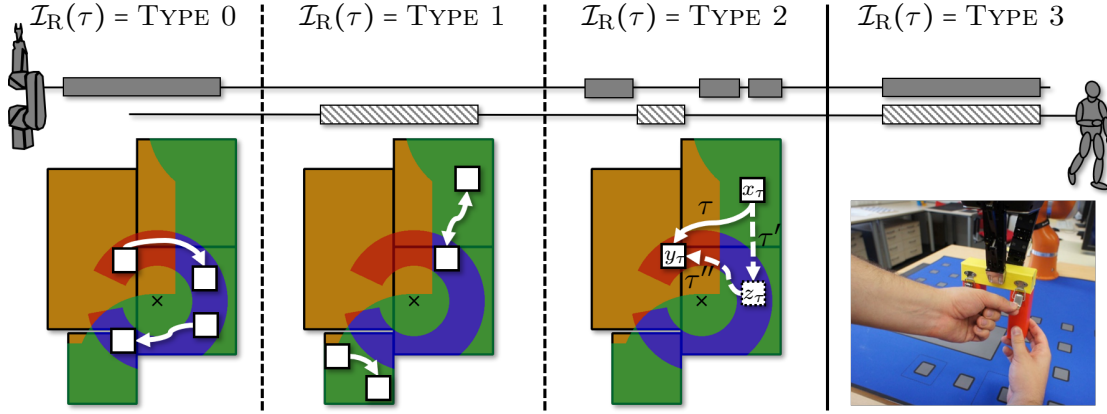


Figure 5.4.: Operations with different interaction categories require different temporal involvement of humans (shaded time blocks) or the robot (solid time blocks). From the robot point of view, only robot action (TYPE 0), only human action (TYPE 1), sequential cooperation (TYPE 2) or synchronous collaboration (TYPE 3) are needed (top). Semantically, the categories correspond to part manipulations in different workspace areas according to the capability model (bottom).

definition of TYPE 1, i.e. $\mathcal{I}_{a'}(\tau)$ must be TYPE 0, if $\mathcal{I}_a(\tau) = \text{TYPE 1}$ in our two-agent case. One of the agents is not involved in such operations. Continuing the example, TYPE 1 applies to situations, where (i) parts are moved within the human-exclusive workspace (ii) or where objects are moved between the human-exclusive and shared regions.

TYPE 2: An operation τ is assigned TYPE 2, if and only if it is neither TYPE 0, nor TYPE 1 from the point of view of agent a , but if it can be achieved in *sequential cooperation*. We define an operation to be feasible in sequential cooperation if it can be decomposed into a sequence of TYPE 0 and TYPE 1 operations. This way, operations that cannot be handled by one agent are broken down into pieces that integrate with decoupled, parallel working: The operation is automatically split and replaced by the resulting parts in the precedence graph. Then, the newly added operations can be handled according to their precedence relations. As shown in Figure 5.4, this leads to a sequence of human and robot operations that result in the original goal condition of τ . For the purpose of this decomposition, we introduce the *decomposition function* $\text{decompose}(\tau)$ as an interface to a symbolic planner, with output according to Equation 5.5.

$$\text{decompose}(\tau) = \begin{cases} \emptyset & \text{if decomposition is infeasible} \\ (\tau', \tau'', \dots, \tau^{(n)}) & \text{otherwise} \end{cases} \quad (5.5)$$

Any underlying planner implementation is intended to split τ into a sequence of n new operations $(\tau', \tau'', \dots, \tau^{(n)})$. In this sequence, $\tau^{(j)}$ precedes $\tau^{(j+1)}$ ($\tau^{(j)} <_T \tau^{(j+1)}$). The interaction categories $\mathcal{I}_a(\tau^{(j)})$ of all components are either TYPE 0 or TYPE 1. In addition, the preconditions of τ' , and the postconditions of $\tau^{(n)}$ must be identical to those of τ . If planning a decomposition with these properties is infeasible, $\text{decompose}(\tau)$ returns the empty set \emptyset . TYPE 2 occurs in the benchmark domain, whenever parts are moved from robot-exclusive to human-exclusive workspace areas or vice versa. For

this special case, the limitation in individual agent capabilities can be resolved by a planner implementation that generates object hand-overs: Let the triple (o_τ, x_τ, y_τ) characterize the operation τ that moves an object o_τ from position $x_\tau \in \mathbb{R}^3$ to $y_\tau \in \mathbb{R}^3$. This characterisation applies to all **Pick&Place** variants in Table 3.2. The decomposition function can then be defined to split TYPE 2 operations according to

$$(o_\tau, x_\tau, y_\tau) \mapsto ((o_\tau, x_\tau, z_\tau), (o_\tau, z_\tau, y_\tau)) \quad (5.6)$$

by planning a hand-over part position z_τ in the shared workspace W_{shared} . As indicated in Figure 5.4, a human agent can then e.g. transfer the part from x_τ into the shared workspace (τ'). Then, the robot can eventually carry out the final transfer to y_τ (τ''). More complex planners underlying the **decompose** interface may be used for other domains based on the basic action inputs and prediction functions within the skill framework.

TYPE 3: The final interaction category covers operations that cannot be classified as Type 0, TYPE 1 or TYPE 2. Such operations are not feasible for any single agent, and they cannot be performed in sequential cooperation. Consequently, TYPE 3 requires tightly coupled co-working. Respective operations can only be executed, when agents work together simultaneously – this interaction category thus applies to all operations involving collaboration according to the definition in Section 1.2. Within the benchmark domain, this is the case for instances of the **Assemble** skill. It is important to notice that TYPE 2 decompositions do not introduce strict synchronisation like TYPE 3: Component operations must be carried out in consideration of their precedence relations. Agents, however, do not need to carry out all operations of a decomposition in a coherent row, but may use their decision authority to fit them suitably into their workflow.

With the above semantics, a general classification scheme for interaction categories per agent directly follows from the capability model:

$$\mathcal{I}_a(\tau) = \left\{ \begin{array}{ll} \text{TYPE 0} & \text{if } C_a(\tau) \\ \text{TYPE 1} & \text{if } \neg C_a(\tau) \wedge C_b(\tau) \\ \text{TYPE 2} & \text{if } \neg C_a(\tau) \wedge \neg C_b(\tau) \wedge \text{decompose}(\tau) \neq \emptyset \\ \text{TYPE 3} & \text{else} \end{array} \right\} \text{ with } \begin{array}{l} a, b \in \{\text{H}, \text{R}\} \\ a \neq b \end{array} \quad (5.7)$$

Specifying the capability model and implementing a planner to **decompose** operations adapts this classification process to concrete domains. Communication patterns (Section 5.1.3) and component team mental models (Section 5.1.4) can then be defined based on the abstract notion of interaction categories. This way modelling robot participation in terms of communication, perception, and action is decoupled from individual use-cases.

5.1.3. Flexible Communication Patterns

Aside from skill execution and visual perception, communication is a crucial capability to enable all levels of coordination for flexible human-robot teaming (Figure 1.3). Robot companions in flexible scenarios may e.g. need to call absent human partners for help with collaborative operations in the course of teaming-level coordination, if no other feasible

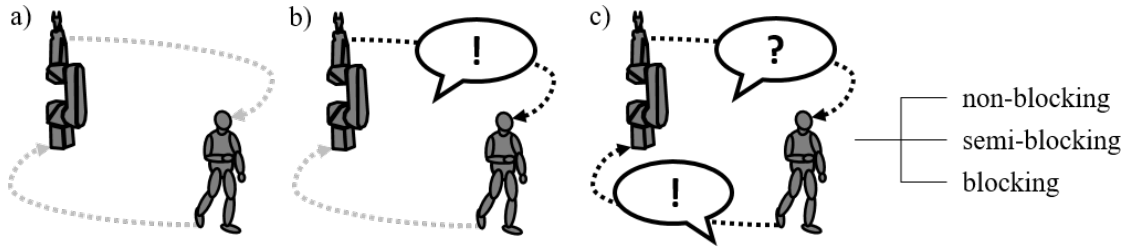


Figure 5.5.: Relevant communication patterns involve robot-to-human and human-to-robot information flow (dashed arrows). Disabling (grey arrows) or activating (black arrows) and sending requests in terms of questions (?) and notifications (!) leads to coordination without explicit communication (a), with one-directional (b) or bidirectional interaction (c).

subtasks are left to take over. When talking about explicit communication, we will rely on the following nomenclature and delimitations: Agents can issue *communication requests* to enter interaction with peers. Generally, a request combines an arbitrarily shaped message with a possibly empty set of conceivable answers. In our benchmark domain, recipients can either *accept* or *decline* a request, as far as providing an answer is required at all. The backward communication channel is thus limited to clear ‘yes’/‘no’-answers. This is sufficient to negotiate mutual commitment or agree on a handover position. It furthermore avoids complex dialogue management, which exceeds the limits of this endeavour. Different communication channel implementations have extensively been studied in prior literature (cf. e.g. [42, 83]), and an asynchronous, smartphone-based messaging app is explained in Section 6.1.1. By contrast, the focus of this section lies on identifying abstract communication patterns that can inspire formulation of suitable team mental models for the robot system. In line with the problem analysis in Section 1.3, such patterns must especially avoid blocking communication to the greatest possible extent – this way, the system is kept capable of acting, even if the recipient of messages is occupied otherwise and does not respond currently. Looking at related literature, we can identify three general communication patterns (Figure 5.5):

No explicit communication channels: Approaches without explicit communication solely rely on sensor data to track task progress and trigger robot action [30, 9, 44]. Information is exchanged implicitly via changes of the world state, or by observation of human actions – yet there is neither a robot-to-human nor a human-to-robot information channel to exchange messages explicitly (Figure 5.5a).

Unidirectional Information: The second communication pattern involves unidirectional information flow issued by one agent. E.g., the robot system may communicate pieces of information to a human partner (Figure 5.5b) without expecting an answer. This pattern is used, e.g. to confirm progress explicitly after finishing operations [60, 112], to inform human partners about the progress they missed during a phase of absence [33], or in the context of more complex schemes for robot feedback [114]. Technically, information through unidirectional notifications can be used by robots in flexible teams at any time, as recipient reactions are not required. Thus, the system cannot be hindered from working, as there is no need for awaiting an answer.

Bidirectional Communication: Figure 5.5c visualizes the final pattern, where communication is bidirectional. One agent requests information by asking a question and expecting the recipient to provide an answer eventually. Bidirectional processes are often used to establish a shared understanding of task allocation, e.g. by incrementally negotiating individual steps [111, 103] or the whole plan [85]. Within this class, three variants can be distinguished: *Blocking communication* means that the requesting agent will stop acting and wait until the required answer has arrived. This busy waiting for answers keeps a robot from being productive in situations where partners do not respond. Blocking communication is thus particularly critical for robot participation in flexible teaming. In the other extreme case of *non-blocking communication*, a message is sent to the recipient, but arriving answers are not necessarily processed immediately. Responses are rather deposited in an inbox that the robot system will check as soon as possible, e.g. after finishing an operation. This procedure avoids blocking but may be undesirable in teaming scenarios: Imagine a situation, where the robot asks its partner to help with a collaborative **Assemble** action. The human team member utters commitment to join in shortly after, but the robot has initiated a different, time-consuming operation in the meantime – this may be unintuitive for humans, as the system has requested help, but is itself not ready to collaborate.

The spectrum is therefore complemented by introducing a pattern called *semi-blocking communication* to be used in such situations [135]. This pattern incorporates blocking communication but aborts the busy waiting loop after a reasonable timespan. Termination of this procedure is guaranteed by attaching a *timer value* $\mathcal{T}_t : r \mapsto [0, 1]$ to each communication request r at time t . Starting from 1 in the moment of issuing the request, this timer value is decremented according to

$$\mathcal{T}_{t+\Delta t}(r) = \mathcal{T}_t(r) - c_p \cdot \frac{1}{1 + |H_t|}, \quad (5.8)$$

with some small time step duration Δt . A request is assumed declined or *timed out* as soon as $\mathcal{T}_t(r)$ reaches 0 – waiting for responses can then be stopped. The calculation of \mathcal{T}_t depends on two magnitudes: (i) The current number of entries $|H_t|$ in the human presence map captures whether workers are currently at the workbench (Section 4.2.2). Assuming that workers are cooperative, a reaction to the request r is likely as long as $|H_t|$ is large. On the contrary, it is reasonable to discard a request faster, if no worker is available. Similar to the idea behind world model ageing, the request timer value thus decays depending on $|H_t|$. (ii) A *request patience constant* $c_p \in]0; 1[$ scales this decay process. For $c_p \rightarrow 0$, $\mathcal{T}_t(r)$ will hardly decrease, and discarding requests when \mathcal{T}_t reaches 0 thus resembles blocking communication. In the other extreme case ($c_p \rightarrow 1$), communication is aborted immediately in situations where $|H_t| = 0$ similar to non-blocking requests. The patience constant must be chosen to give humans an appropriate amount of time to finish their current action and react to the request. It is important to notice that request timers are not exclusively used with semi-blocking, but with robot-issued requests of any type of unidirectional or bidirectional communication. We will see in the following section how a recurring ‘reminder functionality’ in order to stay in touch with workers can be integrated into team mental models this way.

5.1.4. Preemptive State Machines

Individual communication patterns can now be used as templates to design a component team mental model for each interaction category. These models are defined by preemptive state machines and can be understood as the life cycle that operations need to undergo during cooperative execution. A *preemptive state machine* is a hierarchically structured, finite state machine in the sense of Harel’s state charts [45]. The state hierarchy is specifically composed of three state-levels (Figure 5.6): The top level is called the *task progress level*. It holds states matching those of the task progress estimate (Equation 3.22), i.e. INACTIVE, ACTIVE and DONE. The ACTIVE state contains the two sub-states EXECUTE and MONITOR of the *activity mode level*. Semantically, the MONITOR state represents the assumption that another agent than the robot is currently occupied with the associated operation. MONITOR sub-states on the lowest *activity level*, where each state is linked to an actual robot activity, are therefore dedicated to monitoring task progress (e.g. by actively validating conditions). By contrast, child states of EXECUTE are intended to promote execution of the operation, e.g. by executing skills, or by initiating communication. Robot activities are realised by different architectural components of the overall system (e.g. by the Skill Execution Engine) – they furthermore yield events such as ‘Skill execution failed’, ‘Preconditions satisfied’ etc. These events trigger state transitions, thus advancing operations within their life cycles.

The team mental model state machines are generally assumed preemptive, i.e. there is no assumption on the number of state transitions and activities to be done before turning to another operation. Still, state machine semantics is complemented with *non-preemptive activity blocks* that have to be carried out consecutively without interruption. Similar to the motivation of semi-blocking communication, the robot should e.g. reliably fulfil its role in a collaborative operation directly after a partner has agreed to join in. To ensure such sequences, states of the activity mode level can explicitly be marked non-preemptive. Taking these considerations and the communication patterns of Section 5.1.3 in mind, we can now look at concrete state machines used in this work (Figure 5.6). Bearing in mind the major project goals of flexibility and dynamic decision making, they are designed to reduce close interaction and foster decoupled, parallel co-working. Explicit communication is therefore only used where strictly necessary, and blocking requests are avoided in particular.

TYPE 0: Operations of this type can be carried out by the robot. Availability of resources and postconditions can also be checked by moving the camera and detecting parts, as all relevant locations must lie in robot range (Equation 5.3) – TYPE 0 can thus be handled by robots without explicit communication, and coordination is achieved by world state observation only. The state machine thus produces the following behaviour: Preconditions are checked first. If all resources are available, the system can try execution of the underlying skill. These two steps form a non-preemptive block, as checking preconditions, but turning to another operation directly afterwards would be inefficient. Aborted skill execution as well as missing parts during the precondition check indicate that some other agent seems to have interacted with necessary resources. In this case, the operation enters the monitoring phase to check whether the operation has already been

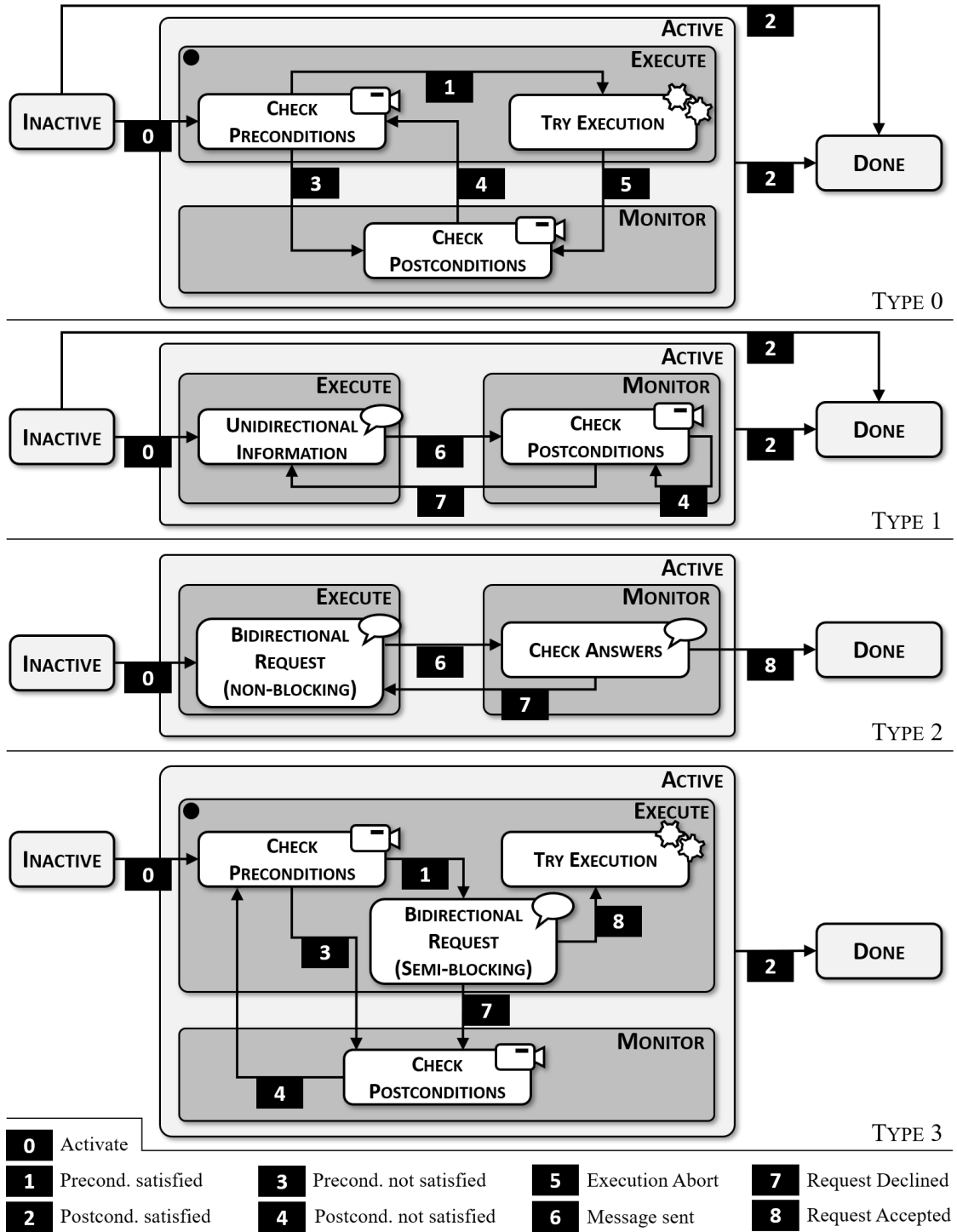


Figure 5.6.: The preemptive team mental model state machines are composed of the task progress level (light grey), the activity mode level (dark grey) and the activity level (white). Activities in non-preemptive states (●) must be carried out consecutively, until an event (■) triggers a transition (→) into another state of the activity mode level.

handled. If the system cannot observe progress, the state changes back to EXECUTE, thus scheduling the operation for a retry. The state machine enters DONE independently from the previous state as soon as the system perceives that postconditions are satisfied.

TYPE 1: The robot cannot handle operations of TYPE 1. But we cannot assume that workers know about this fact – this would require profound worker knowledge on kinematic properties and robot capabilities. The system relies on the pattern of unidirectional information to resolve this issue. As soon as TYPE 1 operations are ACTIVE, the first robot activity sends an explicit notification to the human. This notification transports the information about lacking robot capabilities and clarifies that respective operation falls in the worker’s scope of responsibility. The robot then needs to check postconditions and will eventually detect that the operation is DONE. Although unidirectional information does not require an answer, the state machine covers the case of a declined request. This is because the concept of request timers is also used with unidirectional messages – this way, the request will time out after some time, which triggers a ‘Request declined’-event. The system can now repeatedly remind workers of their responsibilities in the task.

TYPE 2: The state machine for this interaction category has a structure similar to that for TYPE 1 operations. Human and robot co-work based on the shared task mental model, and common ground on the task is a crucial feature of teaming [52]. A TYPE 2 decomposition does, however, mean a violation of this shared understanding, as an operation is replaced by a sequence of new process steps – peers must thus mutually accept the implications of splitting the operation on the task model. This can only be achieved by using the bidirectional communication pattern. The robot will therefore send a bidirectional, non-blocking request. In the case of the benchmark domain, the aim of this request is agreeing on a hand-over position proposed by the robot. The operation can be preempted after sending the message and the system may proceed with other parts of the task while waiting for consent. If the request is accepted, the proposed changes can be applied to the task model. The TYPE 2 operation is then no longer existent in the model and can thus be promoted into the DONE state. A declined request, emerging either from explicit utterance of rejection or from a request timer timeout, causes the robot to initiate a new round of negotiation.

TYPE 3: Collaborative operations involving more than one agent are handled based on the semi-blocking communication pattern. The state machine structure and workflow is similar to that for TYPE 0, but incorporates additional communication to request help. As with TYPE 0, the robot must first confirm availability of relevant resources by checking operation preconditions. A semi-blocking, bidirectional request is issued next. The message sent will prompt peers to join the robot. If any present worker accepts the request, the robot can issue skill execution and fill in its role within the collaborative act. We do here not consider cases where skill execution is aborted – human and robot have agreed on collaborating beforehand, and thus there is no need for assuming failure presuming skilled, cooperative workers. As discussed in Section 3.1, skill graphs may themselves incorporate communication actions. Respective interaction may even follow the strictly blocking communication pattern, e.g. when waiting for the confirmation

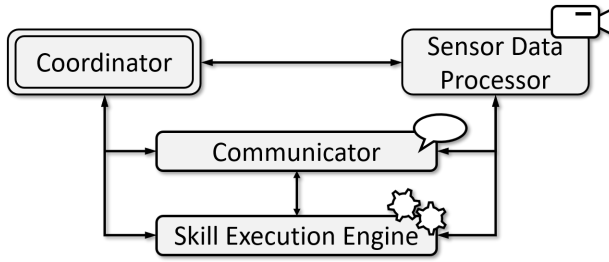


Figure 5.7.: Architecture of the system: The individual components implement activities that are marked correspondingly in Figure 5.6 (skill execution ⚙️, perception 📷 and communication 💬).

of successful assembly during the **Assemble** skill. The TYPE 3 protocol makes sure that such inevitably blocking interaction is always preceded by a non-blocking attempt to establish mutual commitment. Message recipients in this process may decline the request or miss answering before request timeout. The system will then try to unveil by evaluating postconditions whether two humans have joined and handled the operation. After a negative condition check, the protocol is re-iterated.

5.2. System Architecture

The team mental models use activities that belong to either of the three categories skill execution, perception or communication. In consequence, the overall system architecture (Figure 5.7) incorporates one component each to implement the activities of respective groups. The interplay of these assistive components is managed by the *Coordinator*, where robot participation in dynamic task sharing is ultimately achieved by reasoning and knowledge update steps (Section 5.3). In this section, we will look at the assistive components. They have the following functionality and interdependencies:

Skill Execution Engine: The Skill Execution Engine takes operations of TYPE 0 or TYPE 3 as an input, as the robot can participate in execution of such operations during the TRYEXECUTION activity. The engine therefore implements all basic actions for a specific hardware platform. It can thus execute a parametrised skill by repeatedly calling action implementations and branching to the next action based on previous action outcomes (Section 3.1). Implementations of communication actions (Section 3.1.1) construct abstract requests that define the message and communication pattern to be used. Requests are then passed to the Communicator component which maintains a concrete communication channel for transmission. After skill graph traversal, the operation result (success, abortion after reverse execution, or failure) is reported to the commanding component as a state machine event. Operations can emerge from the task model, meaning that execution is directly commanded by the Coordinator. In addition, the Sensor Data Processor may issue perception operations (cf. the **Approach&CaptureImage** skill in Table 3.2) to validate conditions. The system can be adapted to different hardware by changing the implementation of the execution engine.

Sensor Data Processor: The Sensor Data Processor provides two modes of perception. (i) Any motion caused by the Skill Execution Engine moves the eye-in-hand robot camera system through the workspace. In the course of *passive perception*, the stream of

images captured during motion is processed. Each image undergoes an object recognition procedure and the results are integrated consistently into the human-aware world model following the procedure defined in Section 4.1. The passive component also implements world model ageing. If required by the chosen ageing strategy, this incorporates cyclic extraction of human positions from LIDAR data. (ii) The Sensor Data Processor is furthermore a service that provides *active perception* to answer queries concerning the state of operation conditions in `CHECKPRECONDITIONS` and `CHECKPOSTCONDITIONS` activities. To this end, perception operations are planned to approach and monitor relevant points of interest using the `Approach&CaptureImage` skill of the benchmark domain. Active perception is, however, only triggered if the world model does not already confirm all parts in question with a certainty score of 1. Moreover, not all pre- or postconditions must necessarily be perceived. The requested set of conditions is therefore first filtered to reduce the number of perception operations: During a `CHECKPRECONDITIONS` activity, only conditions regarding parts that are actively modified by the operation are evaluated. If there is no basic action within the operation where the prediction function changes the part state, the part will be *passive* during this operation. Existence of passive parts (e.g. the container during `Pick&PlaceInContainer` operations) is generally ensured by precedence relations and prior task progress. These parts need not be checked separately, as the system will not try to manipulate them. By contrast, perception is planned for each processed resource or tool precondition. Similarly, the `CHECKPOSTCONDITIONS` activity perceives only conditions where part output states differ from corresponding input states. Any other objects (e.g. tools that are returned to their initial mounting position) are not informative regarding achieved task progress.

Just as operations within the task model, planned perception operations may be infeasible for the robot. This case occurs particularly for conditions of `TYPE 1` operations, where part locations out of robot reach are involved. Corresponding perception operations are then also classified as `TYPE 1`. The Sensor Data Processor is connected to the Communicator for handling these situations in which the robot system itself cannot observe the parts in question. This way, human partners can be asked to confirm or deny existence of the object states. Respective object states are then inserted into the world model as soon as existence is confirmed.

Communicator: The Communicator realises a concrete communication channel (e.g. the smartphone messaging app outlined in Section 6.1.1) to transmit requests and receive answers. Similar to the Skill Execution Engine, different channels (e.g. for verbal or gesture-based interaction) may be integrated into the system by implementing a Communicator to control suitable hardware components in future experiments. For the domain and team mental models used in this work, any Communicator implementation needs to support the conceptual requests listed in Table 5.1. The component team mental models according to Figure 5.6 involve communication activities for operations of `TYPE 1`, `TYPE 2` and `TYPE 3`. Team mental models are integrated into the Coordinator, and this component thus contributes three request types according to the communication activity semantics in respective state machines. Communication requests are furthermore necessary if the Sensor Data Processor cannot observe some condition itself, and during

	Message	Answers	Pattern
Coordinator			
① TYPE 1 operations	Hint to guide peers towards carrying out respective operations	\emptyset	$\circ \rightarrow$
② TYPE 2 operations	Robot-planned suggestion for a decomposition/hand-over position	{yes, no}	$\circ \leftrightarrow$
③ TYPE 3 operations	Invitation to join in with a collaborative operation	{yes, no}	$\bullet \leftrightarrow$
Sensor Data Processor			
④ TYPE 1 operations	Question to confirm part existence when perception operations would be TYPE 1	{yes, no}	$\circ \leftrightarrow$
Skill Execution Engine			
QUERYCOMPLETION basic actions	Prompt to confirm successful completion of assembly	{yes}	$\bullet \leftrightarrow$

Table 5.1.: Conceptually necessary communication requests defined by message, set of conceivable answers, and communication pattern (\rightarrow = unidirectional, \leftrightarrow = bidirectional, \circ = non-blocking, \bullet = semi-blocking, \bullet = blocking) to be issued by the architectural components when encountering the listed situations

execution of skills with communication actions. For (semi-)blocking communication, function calls to the Communicator are synchronous and will only return after an answer or request timeout. Components issuing non-blocking requests receive so-called *handles* in turn. These handles identify requests and enable querying answers when needed while the Communicator processes the request asynchronously.

5.3. Dynamic Task Sharing

The robot system participates in dynamic task sharing by iterating the conceptual coordination and reasoning loop shown in Figure 5.1. Based on the team mental models and architectural components, this loop is specified more precisely by Algorithm 1: The *initialization phase* prepares the world model and task model for execution. First, all parts and their goal states are inserted into the world model (Line 1) according to Section 4.1. In addition, interaction categories are assigned to the operations of the task model (Line 2). The system can then track the current state of each operation τ in the corresponding team mental model (Figure 5.2): We will refer to the current, most deeply nested state of τ as **state_of**(τ). Recall that states in our preemptive state machines are grouped hierarchically – an operation with **state_of**(τ) = TRYEXECUTION is thus also in the parent states EXECUTE and ACTIVE of TRYEXECUTION (Figure 5.6). The state hierarchy is then accessed by a lookup function **is_in_state**(τ, σ). This function returns TRUE, if and only if the queried state σ is a parent state of **state_of**(τ).

After initialization, the algorithm proceeds with loop iterations, until the system has recognised that all operations are DONE (Line 13). In each of these *working phases*,

the procedure favours operations in the EXECUTE state (Line 4). In line with the state semantics in Section 5.1.4, respective sub-tasks promise robot action that contributes to task progress. If no matching operations are left, the current loop iteration is dedicated to perception of progress achieved by other agents (Line 7). In both cases a subset $T' \subseteq T$ of operations for either execution or monitoring can be extracted. The subset in question is then passed to the `advance_task()` function. Section 5.3.2 describes how this function selects an operation from T' and issues activities according to the team mental model. The task allocation decision is based on different strategies $\mathcal{W}_{\text{EXECUTE}}$ and $\mathcal{W}_{\text{MONITOR}}$ – these strategies are encoded in different functions to calculate weights for each operation, thus introducing priorities according to their likeliness of success (Section 5.3.1). Communication is not considered as a dedicated phase in the algorithm, as related activities are embedded into the EXECUTE as well as MONITOR states.

Algorithm 1 Reasoning and Coordination Loop

```

1: initialize_world_model()                                ▷ Initialization
2: determine_interaction_categories()

3: repeat

4:   if  $\exists \tau \in T : \text{is\_in\_state}(\tau, \text{EXECUTE})$  then      ▷ Working
5:      $T' \leftarrow \{\tau \in T \mid \text{is\_in\_state}(\tau, \text{EXECUTE})\}$ 
6:     advance_task( $\mathcal{W}_{\text{EXECUTE}}, T'$ )
7:   else
8:      $T' \leftarrow \{\tau \in T \mid \text{is\_in\_state}(\tau, \text{MONITOR})\}$ 
9:     advance_task( $\mathcal{W}_{\text{MONITOR}}, T'$ )
10:  end if

11:  update_progress()                                       ▷ Knowledge Update
12:  update_communication()

13: until  $\forall \tau \in T : \text{is\_in\_state}(\tau, \text{DONE})$ 

```

Each loop iteration ends with a *knowledge update phase* (Line 11). This phase advances operation states based on perception results gathered during motion in the preceding working phase. Furthermore, messages that have meanwhile arrived are processed. Details on this process are given in Section 5.3.3.

5.3.1. Decision-Making Strategies

The system makes task allocation decisions based on decision-making strategies. Generally, a *decision-making strategy* is defined as a weighting function $\mathcal{W} : T \rightarrow \mathbb{R}_0^+$ on the set of operations T . Decision-making strategies are intended to produce high function values for operations that the robot should engage in and low values for those that

are not promising at the moment. This prioritisation process makes the overall system configurable: Decisions mean choosing the operation with the highest numerical priority value. Concrete criteria underlying realisations of \mathcal{W} can then be adapted arbitrarily to consider aspects as outlined in Section 2.2. Algorithm 1 uses two strategies $\mathcal{W}_{\text{EXECUTE}}$ and $\mathcal{W}_{\text{MONITOR}}$ – this is because decisions for active participation may require different criteria (e.g. preferences or trust) than those for monitoring (e.g. the information gain). This work seeks to enable parallel working under partial workspace observability. Strategy definitions therefore strongly incorporate the notions of readiness $\mathcal{R}(\tau)$ and success $\mathcal{S}(\tau)$ of an operation τ as introduced in Section 4.3. These metrics produce likelihood values in the interval $[0, 1]$ to judge whether an operation is likely to succeed (\mathcal{R}) or has probably already been done (\mathcal{S}), considering possibly outdated information from the world model. Adding to the authors prior work [134] on action selection, examples of strategies based on these metrics are defined as follows:

We know the current number of humans $|H_t|$ near the workbench for a decision at time t . This number is assumed a constant per decision. With $|H_t|$ given, we can introduce a *necessity function* $\nu : T \rightarrow \{0, 1\}$, with

$$\nu(\tau) = \begin{cases} 1 & \text{if } |H_t| > 0 \wedge \mathcal{I}_H(\tau) = \text{TYPE } 1 \\ 0 & \text{otherwise} \end{cases}. \quad (5.9)$$

This function enables judging whether handling τ is of particular importance for a parallelised human-robot workflow. It results in a value of 1, if τ can only be performed by the robot, i.e. if $\mathcal{I}_H(\tau) = \text{TYPE } 1$ from the human point of view. Preferring operations like these is however only necessary if workers are present at the moment ($|H_t| > 0$). Based on ν , we can construct a decision strategy $\mathcal{W}_{\text{EXECUTE}}$ that seeks to minimize erroneous robot operation attempts while trying to complement human skills proactively. The weighting function is given by

$$\mathcal{W}_{\text{Execute}}(\tau) = \begin{cases} \mathcal{R}(\tau) + \nu(\tau) & \text{if } \mathcal{I}_R(\tau) \in \{\text{TYPE } 0, \text{TYPE } 3\} \\ \infty & \text{otherwise} \end{cases}. \quad (5.10)$$

This definition distinguishes between operations in which robots can take an active role in part manipulation (TYPE 0 and TYPE 3), and those merely involving communication activities (TYPE 1 and TYPE 2). For the latter group, $\mathcal{W}_{\text{EXECUTE}}$ assigns an infinite weight. Notifications about TYPE 1 operations and negotiations of TYPE 2 decompositions are inevitable. They can thus be handled with maximum priority. Other operations are rated depending on the likelihood \mathcal{R} of part availability. This particularly integrates the ergonomic considerations behind the definition of interaction indicators into the decision process (Section 4.2.2). The necessity function value for τ amplifies the resulting weight for operations that only the robot can do, i.e. where $\mathcal{I}_H(\tau) = \text{TYPE } 1 \wedge \mathcal{I}_R(\tau) = \text{TYPE } 0$ holds. This applies particularly to part transfers from shared to robot-exclusive workspace areas in pick-and-place scenarios: Relevant parts are here initially located in shared areas within human reach. This will likely result in low readiness scores $\mathcal{R}(\tau)$ when workers are present. Without the amplification by $\nu(\tau)$ respective operations would thus receive

an overall low weight $\mathcal{W}_{\text{EXECUTE}}(\tau)$, although they must be done by the robot in any case. Adding the necessity function value thus biases decisions towards preferring operations that peers are incapable of.

The strategy $\mathcal{W}_{\text{MONITOR}}$ distinguishes between TYPE 2 operations and those of any other interaction category. Negotiating a decomposition removes any TYPE 2 operation from the task model – thus there is no need to monitor progress at all, i.e. these operations can be assigned the lowest possible weight of 0. For all other operations high success values $\mathcal{S}(\tau)$ indicate likely manipulations in spatial areas around their goal states – issuing perception towards these locations is thus promising to uncover progress achieved by one’s peers. Therefore, it is reasonable to calculate weights for monitoring according to

$$\mathcal{W}_{\text{Monitor}}(\tau) = \begin{cases} 0 & \text{if } \mathcal{I}_R(\tau) = \text{TYPE 2} \\ \mathcal{S}(\tau) & \text{else} \end{cases}. \quad (5.11)$$

5.3.2. Task Advancement

Given a decision strategy \mathcal{W} and a set of candidate operations T' , Algorithm 2 issues the necessary robot activities to advance one of these operations in line with the matching state machine. First, the operation $\tau_{\text{opt}} \in T'$ with maximum weight is selected (Line 3).

Algorithm 2 Task Advancement Procedure

```

1: procedure advance_task( $\mathcal{W}, T' \subseteq T$ )
2:    $x \leftarrow \text{FALSE}$  ▷ Stopping condition
3:    $\tau_{\text{opt}} \leftarrow \arg \max_{\tau \in T'} \mathcal{W}(\tau)$  ▷ Task Allocation Decision

4:   while  $\neg x \wedge \neg \text{is\_in\_state}(\tau_{\text{opt}}, \text{DONE})$  do

5:      $\sigma \leftarrow \text{state\_of}(\tau_{\text{opt}})$  ▷ Task Advancement
6:      $\epsilon \leftarrow \text{delegate\_activity}(\sigma)$ 
7:      $\text{advance\_state}(\tau_{\text{opt}}, \epsilon)$ 
8:      $\sigma' \leftarrow \text{state\_of}(\tau_{\text{opt}})$ 

9:     if  $\text{is\_preemptive}(\sigma)$  then ▷ Evaluate stop criteria
10:       $x \leftarrow \text{TRUE}$ 
11:     else if  $\text{activity\_mode\_changed}(\sigma, \sigma')$  then
12:       $x \leftarrow \text{TRUE}$ 
13:     end if

14:   end while
15: end procedure

```

If several operations with identical, maximum weight exist, the decision among them is taken randomly. Having finished this task allocation decision, steps according to the

mental model state machine matching $\mathcal{I}_R(\tau)$ can be taken. Flexible teaming requires a responsive mode of robot decision making: It is thus not always desirable to work on an operation consecutively until the DONE state is reached. Consider e.g. a situation where a worker leaves the workbench and takes a tool with him accidentally. The system might then make an unfavourable task allocation decision for an operation involving this tool. For a TYPE 0 operation, this would e.g. result in repeated search for the tool (CHECKPRECONDITIONS) and checking for task progress (CHECKPOSTCONDITIONS). The system would spin between two states in a livelock-alike manner [6], until the tool is eventually returned to the workbench. The task advancement procedure avoids such situations by triggering as few activities as possible. Control is then frequently passed back to the main coordination loop (Algorithm 1), resulting in a new task allocation decision based on recent knowledge updates.

The loop body to follow transitions in mental model state machines (Line 5) issues at least one activity. To this end, the activity matching the current activity level state σ of the chosen operation τ_{opt} is first delegated to the corresponding architectural component. This component processes the request and returns an event ϵ depending on activity outcomes. With this event, the next state σ' of τ_{opt} can be determined and stored by `advance_state(τ_{opt} , ϵ)`. The `advance_state` function furthermore adjusts the world model in line with the event via the interface for reasoning components (Section 4.1, page 55). The world model can then reliably reproduce evident changes without further perception by consistently adding or deleting parts. In particular, parts that must have gone missing may be deleted after an ‘Execution Abort’ event. Successful skill execution raises the ‘Postconditions satisfied’ event. For this case object states matching the postconditions are inserted, and those matching preconditions can be removed. This also leads to the removal of w-entities indicating the desired goal state of τ_{opt} .

Responsiveness is achieved by issuing only one single activity whenever possible. Consequently, loop execution is aborted immediately after one step, if the initial state σ of τ_{opt} on the activity mode level is preemptive. If, by contrast, σ is explicitly marked non-preemptive, another loop iteration follows. Task advancement is then continued until a transition into another activity model level state is encountered. The helper function `activity_mode_changed(σ, σ')` provides this information by checking particularly whether a crossover from `is_in_state(τ_{opt} , EXECUTE)` to `is_in_state(τ_{opt} , MONITOR)` or vice versa has happened after changing the state of τ_{opt} from σ to σ' .

5.3.3. Knowledge Update

A knowledge update step follows any call to `advance_task` in Algorithm 1. This step synchronises task progress estimates with world model data and information from communication: Any activities involving robot motion may result in changes to the world model, particularly those triggering active validation of operation conditions. Furthermore, answers to prior, non-blocking requests can arrive during subsequent task advancement steps – these changes in knowledge must be brought in line before making any further decisions. An example is shown in Figure 5.8. Since the last update step, the human partner has acknowledged an asynchronous decomposition request for a TYPE 2 opera-

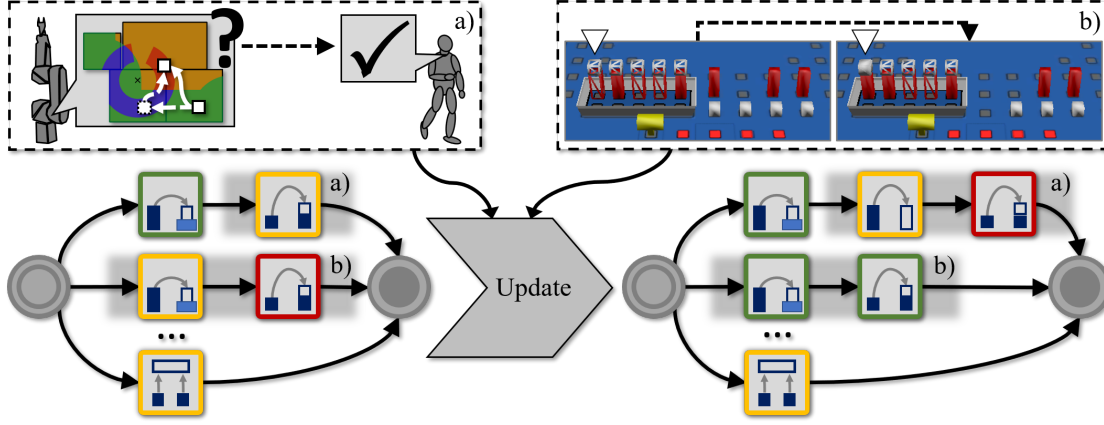


Figure 5.8.: The knowledge update step processes incoming communication and world model changes that occurred during the preceding working phase (\dashrightarrow). Answers to TYPE 2 decomposition requests change the task model structure (a). Newly detected parts (∇) result in an updated task progress estimate (b) with green = DONE, yellow = ACTIVE and red = INACTIVE.

tion (a). The robot can thus adjust its internal task model structure by applying the decomposition. Furthermore, passive perception during motion in the preceding working phase has revealed two parts in the container (b). This leads to an updated task progress estimate, whenever newly detected parts satisfy postconditions of operations. Respective operations can then be considered DONE, which activates subsequent operations. Based on these considerations, we distinguish between two knowledge update phases in Algorithm 1, Lines 11 - 12:

Task Progress Update

The *task progress update phase* ensures correct task execution according to the principles outlined in Section 3.3. Particularly, the robot system may only consider an operation DONE if the existing entity set E_t^e of the world model as defined in Section 4.1 contains object states confirming the postconditions at update time t . Furthermore, operations can only be DONE if all preceding operations have also been completed. Progress on all predecessors may, however, not have been perceived as a consequence of partial observability yet – success may even be unobservable due to occlusions (e.g. after stacking several parts). Nevertheless, their completion can be assumed under the cooperative worker assumption as soon as subsequent progress is confirmed by the world model. Moreover, any newly DONE operation may enable execution of successor graph nodes. This requires triggering ‘Activate’ state machine events. Algorithm 3 keeps task and world model knowledge consistent under these boundary conditions.

The algorithm iterates all operations that have not been done yet. An update is necessary if postconditions are satisfied. To this end, the `postcond_satisfied(.,.)` function matches object-related parameters of the underlying skill of $\tau \in T_{\text{DONE}}$ against known object states in E_t^e . The function returns TRUE if and only if the conditions

specified by Equation 3.21 are TRUE, i.e. if the operation goal state was observed. Any operation τ depends on the completion of all predecessors. By implication, assigning the DONE state to τ also gives information on those preceding operations. Therefore, τ and its predecessors are merged into one set T' . The set elements are brought into a valid execution order in Line 6. A *valid execution order* $(\tau'_1, \dots, \tau'_k, \dots, \tau'_{|T'|})$ is an operation sequence that respects all precedence relations, i.e. if $\tau'_i <_T \tau'_j$, then $i < j$ must hold for all indices $i, j \in \{1, \dots, |T'|\}$. This ordering is established by the helper function `execution_order(.)`. Calling `advance_state` for the elements of T' in the resulting order emulates execution of all operations including τ . The ‘Postconditions satisfied’ event transfers the operations in question into the DONE state (Figure 5.6). Changes to the world model issued by `advance_state(.)` moreover lead to a simulation of incremental part modifications in line with the task model (Section 5.3.2). This adds the partial goal state that must be present after doing τ and its predecessors to the world model.

Algorithm 3 Task Progress Update Procedure

```

1: procedure update_progress()
2:    $T_{\neg \text{DONE}} \leftarrow \{\tau \in T \mid \neg \text{is\_in\_state}(\tau, \text{DONE})\}$ 

3:   for all  $\tau \in T_{\neg \text{DONE}}$  do

4:     if postcond_satisfied( $\tau$ ,  $E_t^e$ ) then
5:        $T' \leftarrow \{\tau' \in T_{\neg \text{DONE}} \mid \tau' <_T \tau\} \cup \{\tau\}$  ▷ Extract predecessors

6:        $(\tau'_1, \tau'_2, \dots, \tau'_{|T'|}) \leftarrow \text{execution\_order}(T')$  ▷ Emulate predecessors for
world model maintenance

7:       for  $i = 1, \dots, |T'|$  do
8:         advance_state( $\tau'_i$ , ‘Postcond. satisfied’)
9:       end for

10:      for all  $\tau'' \in \text{direct\_successors\_of}(\tau)$  do ▷ Activate successors
11:        if  $\nexists \tau''' <_T \tau'' : \neg \text{is\_in\_state}(\tau''', \text{DONE})$  then
12:          advance_state( $\tau''$ , ‘Activate’)
13:        end if
14:      end for

15:    end if

16:  end for
17: end procedure
    
```

After this step, we can look at the successor nodes that represent operations following directly upon τ in the precedence graph. Any subsequent operation τ'' becomes ACTIVE, if it does not possess any predecessor aside from τ that is not yet DONE. If this require-

ment is met, we can call `advance_state(τ'' , 'Activate')` to activate the operation. This procedure is repeated for all elements of T_{DONE} to construct a task progress estimate reflecting all information provided by recent world model content. Vice-versa the world model is complemented with information that follow from the progress update by implication. We have previously used the task progress function P_t for world model ageing in Section 4.2.2. Due to the dualism of possible P_t function values (Equation 3.22) and activity mode level states, $P_t(\tau)$ at time t is directly given by the current activity mode state, i.e. after an update at time t , $P_t(\tau) = \text{DONE}$, if `is_in_state(τ , DONE)` etc.

Communication Update

The *communication update phase* keeps track of non-blocking communication requests issued for TYPE 1 and TYPE 2 operations. In both cases peers are left some time to react, while the robot system may handle other operations. By contrast, TYPE 3 communication does not need to be considered here – respective requests are semi-blocking and must be answered directly within the communication activity. Control is thus not passed back to the coordination loop before the interaction has ended. If TYPE 1 or TYPE 2 requests have been issued, the `update_communication()` function checks all request inboxes for answers in each coordination loop cycle. Answers are treated as follows: (i) TYPE 1 requests are notifications that cannot be answered explicitly. However, they are declined automatically by the Communicator component as soon as the corresponding request timer runs out. If a declined TYPE 1 request occurs for some operation $\tau_{\text{TYPE 1}}$, a 'Request declined' event is raised, i.e. `advance_state($\tau_{\text{TYPE 1}}$, 'Request declined')` is triggered by the update function. (ii) Type 2 negotiation requests can be accepted or declined by the recipient, or eventually result in a timeout. Analogous to TYPE 1, a matching event is raised for the operation in question. In the case of acceptance, the update procedure additionally adjusts the task model (Figure 5.8). After that, an 'Activate' event is raised for the first element of the decomposed sequence.

5.4. Conclusions

Summary

This chapter has shown how a robot system can participate in co-working of flexible teams. The approach relies on the precedence graph task model (Chapter 3) for task progress tracking and perception planning and on the human-aware world model (Chapter 4) for task allocation decisions. The system generally distinguishes between operations according to their interaction needs: An operation may be fully feasible for the robot and thus require no explicit communication with peers. But it may also be infeasible for the robot, require sequential or even synchronous collaboration – in these situations the system may need to access a communication channel. Individual cases are distinguished automatically by assigning one out of four interaction categories (Section 5.1.2) to each operation. This classification pattern relies on agent capability models for humans and robots (Section 5.1.1). A preemptive state machine per interaction category approximates the robot

team mental model. These state machines specify the necessary course of actions for operations in different categories from the robot point of view (Section 5.1.4). To this end, they logically group action, perception and communication activities to execute skills, perceive the state of conditions or communicate with peers. E.g., preconditions must be checked before triggering skill execution, failed skill execution indicates resource allocation by other agents and leads to progress monitoring etc. Where needed, state machines involve particularly flexible communication patterns to avoid blocking interaction, which might otherwise lead the system into an unproductive idle situation while awaiting answers (Section 5.1.3). Task progress is tracked by storing the current state within the corresponding state machine for each operation in the task model. Robot participation is then achieved by repeatedly selecting a promising operation (Section 5.3.1), issuing activities to advance this operation according to the state machine interaction pattern (Section 5.3.2) and reasoning on newly gathered knowledge (Section 5.3.3). All in all, the approach features robot capabilities to (i) participate actively in the task by skill execution and monitoring of task progress, (ii) make peers without expert knowledge aware of sub-tasks that exceed robot capabilities, (iii) negotiate task model changes whenever an operation that no peer is fully capable of can be achieved in sequential cooperation, (iv) establish mutual commitment to initiate synchronous collaboration when needed.

Discussion

The system capabilities for action, perception and communication are each realised by one architectural component (Section 5.2). These components offer implementations for abstractly formulated activities that can be triggered by a central planning and reasoning module. Component implementations are thus decoupled from reasoning algorithms – supplying the system with different implementations enables adaption and extension towards different domains, sensors, actuators or communication channels. The planning module is also modularly designed: It is based on exchangeable task allocation metrics and state machines for team mental modelling. In contrast to learning-based approaches, these state machines are deterministic and explainable. They can be adjusted to change system behaviour by rearranging or adding activities. With regard to the motivation of this work, their current design favours decoupled, parallel working with little communication for enhanced efficiency. Modified task allocation schemes and team mental models open possibilities for future comparative studies on different teaming modes and hybrid team efficiency.

From an algorithmic point of view, the planning layer is highly responsive (Algorithm 1). It keeps the robot capable of acting by frequent task allocation decisions, even if human peers leave the workbench or do not react to communication requests. This is achieved by frequent preemption when performing activities to advance different operations (Algorithm 2). Communication is generally non-blocking or guaranteed to terminate after a predefined, finite amount of time (Section 5.1.3). Blocking communication can only occur as a part of collaborative skills when issuing a TRYEXECUTION activity – execution of such skills is however always preceded by lock-free communication of mutual commitment. Under the cooperative worker assumption, we can further assume that a supportive peer

will stick to this commitment. We have furthermore seen in Section 3.4 that skills do not involve any deadlock-prone waiting for resources. As with all other activities, TRYEXECUTION can thus be assumed to terminate within a reasonable, bounded amount of time. In conclusion, the system is capable of re-planning and acting at any time independent of human presence, as long as one of the remaining, not yet finished operations are feasible for the robot. Temporary livelock-alike situations may occur under certain, seldom circumstances: Consider e.g. a situation where all feasible operations that are currently ACTIVE require a tool that has accidentally been removed from the workbench. The system will then alternate between searching for this tool and trying to detect progress on respective operations – there is currently no designated mechanism to recognize and handle this situation, e.g. by asking to return the tool. Productive work cannot be resumed until further operations are activated or the tool can be accessed.

Currently, communication is exclusively started on robot initiative. The team mental models treat human peers as mostly passive responders and providers of information that the system cannot gather itself with sensors. Contact is only established when strictly necessary. This is motivated by non-expert users' lack of knowledge on robot kinematics and capabilities: Communication is more purposive and efficient, if e.g. the system informs about infeasible operations and plans TYPE 2 decompositions that will surely succeed. This way, time-consuming negotiation of potentially infeasible hand-over positions made up by humans can be avoided. Despite these efficiency considerations for flexible teams, this design also leads to limitations. In particular, collaborative operations can only be initiated on robot request – human peers cannot decide to handle them at will. Truly equal partnership might benefit from fully bidirectional communication initiative. Enhanced possibilities that enable humans to realise their plans, e.g. by deciding to treat collaborative sub-tasks before leaving the workbench, are left for future work. Closely related to this is the issue of cognitive workload. Operations that require communication are favoured during task allocation, as communication is necessary and inevitable. This can lead to situations in which users are confronted with a larger number of requests at a time. Compared to this solely technically motivated strategy, adjusted task allocation metrics or enhanced Communicator implementations can be used to implement more natural communication behaviours in the future. This topic, however, is beyond the scope of this thesis.

CHAPTER 6

Evaluation

6.1. Subjective Evaluation	94
6.1.1. Hardware Prototype	94
6.1.2. Results	97
6.2. Objective Evaluation	99
6.2.1. Benchmark Tasks	99
6.2.2. Simulation System	101
6.2.3. Parametrisation	102
6.2.4. Evaluation Metrics and Reference Data	103
6.2.5. Results	105
6.3. Conclusions	112

THE central hypothesis of this work is that end-users without extensive expert knowledge can operate a system that leads to beneficial, dynamic human-robot teaming, even with limited sensor use. Chapters 3, 4 and 5 have introduced a technical concept for robot participation in this kind of teaming. In this chapter, different aspects of the technical concept are evaluated to investigate the validity of the working hypothesis (Section 1.1, page 5). Hoffmann distinguishes between subjective and objective metrics for evaluating human-robot shared activities [51]. *Subjective metrics* try to capture human perception of the quality of interaction with the robot. They are typically gathered with questionnaires in human subject studies. In extension to this notion, Section 6.1 outlines results on human experience with the *overall* system rather than only the teaming phase. This includes the stages of calibration and task modelling to investigate overall usability. By contrast, *objective metrics* as e.g. task duration, agent idle times etc. can be measured directly and objectively. With this thesis having a clearly technical focus, objective evaluation provides a major part of experimental results (Section 6.2). Performance according to such metrics depends on the robot system behaviour which is in turn influenced by system parameters, by the concrete task and the human working strategy. A comprehensive coverage of these aspects is feasible with a simulation approach that can gather data for an arbitrary amount of benchmark tasks, system parameter sets and

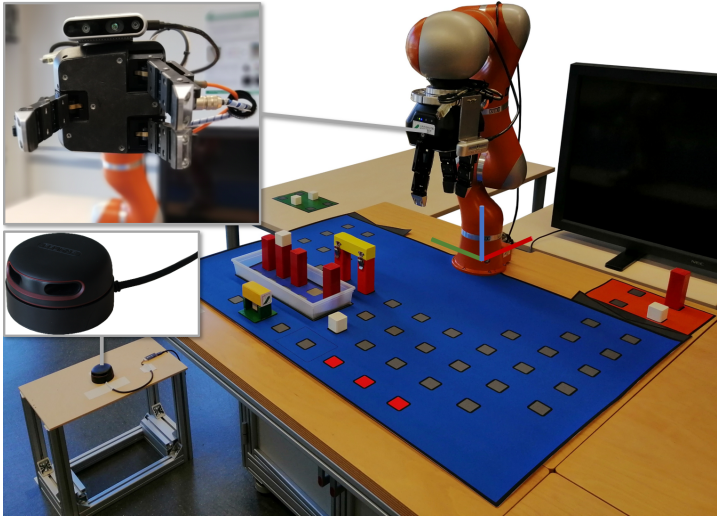


Figure 6.1.: The prototype system setup uses a lightweight robot, a three-finger gripper and an RGB-D camera positioned near the robot tool centre point. A LIDAR sensor is mounted at knee height close to the workbench. Workspace layouts with shared (blue), human-exclusive (green) and robot-exclusive areas (orange), and with dedicated handover positions (red) structure the environment around the world frame (coloured axes) in line with the task model editor.

resulting teaming workflows. This preparatory step of enabling technologies evaluation is suitable for technical advancements prior to future detailed human subject studies [116] – to this effect, lessons learned from the experimental results and future prospects are finally discussed in Section 6.3.

6.1. Subjective Evaluation

Subjective evaluation metrics measure perceived human experience with a system. This requires human subjects to work with a concrete system implementation. The software toolchain and laboratory setup as used for this work is outlined in Section 6.1.1. Implementation details show in particular, how abstract interfaces used in the technical concept can be realised practically. Based on this hybrid workplace realisation, Section 6.1.2 summarises the author’s prior results on end-user operation of the full system.

6.1.1. Hardware Prototype

The prototype setup is shown in Figure 6.1. The system is based on a KUKA LBR 4+ lightweight robot with seven degrees of freedom. A Robotiq 3-Finger Adaptive Gripper in pinch mode enables grasping the small benchmark domain parts (Figure 3.5). Sensory input is provided by a Intel Realsense D435 RGB-D camera. The camera is configured to output structured point clouds with a resolution of 620×350 pixels. Furthermore, human presence data is gathered with a Slamtec RPLIDAR A2 laser range finder. Both sensors are consumer products that can be acquired with a low budget. A workspace layout aligned with the robot base segment (Section 3.2.1) bridges the gap to task modelling in the graphical editor. Humans are enabled to communicate with the system via an Android smartphone application. This section provides implementation details for each architectural component (Section 5.2) with regard to this concrete workplace setup.

Skill Execution Engine

The Skill Execution Engine implements all actions of the example domain (Section 3.1.3). Robot motion is realised by position control on a fixed transfer level that is only left for picking and placing. The tool centre point (TCP) is kept in a fixed orientation in parallel to the workbench surface – camera images are hence taken from a top view with a ‘look-at’ direction perpendicular to the tabletop. The engine moreover implements the **decompose** function to split operations for sequential cooperation during the interaction category classification (Equation 5.5). To this end, the workspace layout provides designated handover positions (Figure 6.1). On request, the Skill Execution Engine generates a decomposition by randomly setting the handover location z_τ for τ (Equation 5.6) to one of these positions. This position is blocked for subsequent decomposition requests and will be released when all operations of **decompose**(τ) are done.

Sensor Data Processor

Sensor Calibration: The system has two sensors that need calibration. Firstly, the robot TCP pose $K \in \mathbb{R}^{4 \times 4}$ is given by robot kinematics equations (Figure 6.2a). In order to represent parts perceived by the eye-in-hand camera in the coordinate frame of the robot world model, we additionally need to know the homogenous extrinsic calibration matrix $X \in \mathbb{R}^{4 \times 4}$ between TCP and camera frame. This problem of eye-in-hand camera calibration is well-known [113] and commonly solved by observing a calibration pattern from different camera poses. After calculating the pattern position in each of the resulting calibration images, X can be determined by optimization, e.g. using the dual quaternion approach of Daniilidis [29]. The prototype offers a one-click software solution for this calibration step. Similar to workspace layouts, the user aligns a calibration pattern with the robot base. The robot will then move the camera to predefined poses and capture calibration images. Afterwards, the optimization starts and stores the extrinsic matrix.

We furthermore need to know about human positions in the world coordinate frame for world model ageing. The coordinate transform $Y \in \mathbb{R}^{4 \times 4}$ can be calculated by applying the approach of Zhang and Pless [131] to calibrate the laser range finder with respect to the world frame. This method requires placing a calibration plate at different positions in view of the camera while also being sensed by the LIDAR sensor. This leads to correspondence pairs of a line in LIDAR data and a plane spanned by the pattern in camera images. These pairs can again be used to optimize the calibration matrix. Figure 6.2b shows the tool that guides users to collect the necessary point-plane correspondences. The interface enables repositioning of the robot so that camera and LIDAR sensor can detect the pattern simultaneously. Correct pattern positioning in sight of both sensors is indicated by a pattern coordinate frame (left) and a line fitted into LIDAR data (right). While moving the pattern the application gathers correspondences, solves for Y and projects LIDAR samples into the camera image (green samples). Users may stop this process as soon as the re-projected samples are seen to cover their legs and the calibration plate sufficiently precise. A more detailed explanation of this two-step camera and LIDAR calibration procedure can be found in the author’s prior work [140].

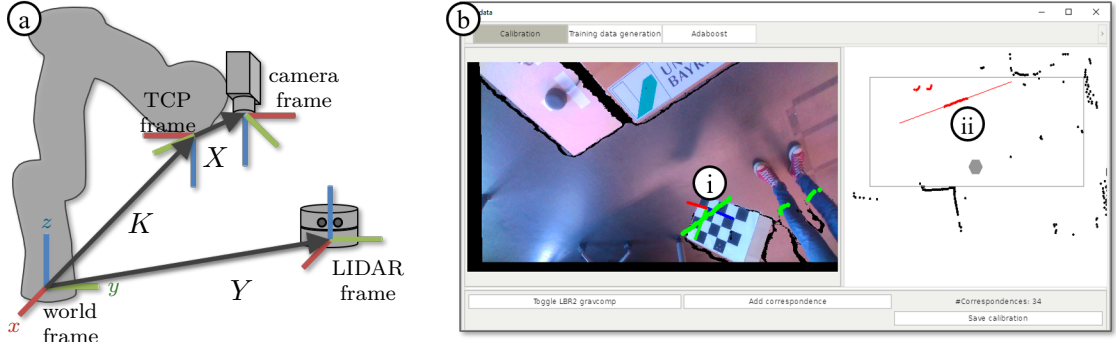


Figure 6.2.: System calibration requires estimating the extrinsic camera calibration matrix X and the transformation Y between world and LIDAR coordinate frame based on the TCP pose K (a). The calibration tool (b) displays the camera image (left) and current LIDAR data (right). A coordinate frame (i) and a fitted line (ii) indicate correct placement of the calibration plate.

Object Recognition: The parts of the benchmarking domain can clearly be distinguished by their colour and dimensions (Figure 3.5). This enables fast and robust object recognition. We know the transformation between robot tool centre point and eye-in-hand camera coordinates from the calibration step. The tool centre point pose in the world coordinate frame is furthermore given by the kinematic properties. The point clouds provided by the camera system can thus be transformed into world coordinates. Given that the world coordinate origin coincides with the mounting point of the robot on the workbench, this point and the world frame x - and y -axis define the workbench surface plane (Figure 6.1). Points representing this plane can then be cropped from the point cloud in a first processing step. The remaining points are split into segments in consideration of mutual spatial proximity and colour using DBSCAN clustering [36]. A bounding box, aligned with the world coordinate frame axes, is determined for each segment. The Sensor Data Processor then removes the points of all object side faces – only points of the top face with a z coordinate value close to the maximum bounding box z coordinate remain in the segments. We can now determine the median colour and extents in the x - and y -direction of the object top faces. These features are fed into a decision tree, which outputs one of the object types \mathcal{T} (Equation 3.15) per segment.

In addition to a robust object recognition procedure, world model updates are based on estimates of part occlusion within the viewing frustum. The viewing frustum planes are directly given by the camera origin in the world coordinate frame, and by technical specifications regarding the opening angle. The `inFrustum(e)` function returns TRUE for some entity e of the world model, if and only if all vertices of the bounding box of e lie within the pyramid spanned by the frustum planes. For world model entries within the viewing frustum, the Sensor Data Processor implementation estimates occlusion of parts on the current camera image as follows: In line with the above object recognition implementation that classifies parts according to their top surface, a world model entity is assumed occluded, if and only if its top face cannot be assumed part of the current point cloud. To this end, the centroid \bar{p}_e of the entity bounding box top face is first determined.

If \bar{p}_e lies below the surface sampled by the current point cloud, `isOccluded(e)` must return `TRUE`. Exploiting the fact that the point clouds are structured images with a pixel grid helps rendering this step computationally efficient. Using the camera extrinsic and intrinsic calibration matrices, \bar{p}_e can be re-projected into the image. This results in the pixel coordinates, where \bar{p}_e would be projected along the corresponding camera eye ray if it was currently visible. We can then compare \bar{p}_e with the actually measured point q in this pixel. Occlusion of e (`isOccluded(e) = TRUE`) is assumed in two cases: (i) The point cloud may not contain a valid measurement for the pixel in question. This occurs regularly, when the eye ray from \bar{p}_e to the camera crosses a surface of the physical world with a steep angle against the ray direction. With our block-like parts and the camera top view, this happens whenever further parts are stacked upon e . The sensing principle limits correct measurements in this case – however, we can still implicate that sight of the top face of e is hindered by some part above. (ii) By contrast, a valid point measurement q in the corresponding pixel confirms occlusion actively, if this point lies closer to the camera origin than \bar{p}_e . If neither condition (i), nor condition (ii) applies, the part e in question should be visible, and `isOccluded(e)` thus returns `FALSE`.

Human Tracking: The prototype implementation uses background subtraction to detect human legs in LIDAR scans. Human person hypotheses are formed by gathering clusters according to the rules described by Topp and Christensen [121]. Legs may temporarily be occluded, e.g. by one’s own other leg during motion or by static obstacles in the workspace (e.g. table legs) – hypotheses are therefore fed into a multi-target tracker based on the approach of Schulz et al. [110] to keep track of persons in such situations.

Communicator

The prototype Communicator implementation connects to an Android application running on workers’ smartphones. Inspired by traditional messengers, this application presents a list of messages related to recent requests to the user (Figure 6.3). Blocking requests are particularly highlighted to stress that the system urgently needs an answer to continue working. The Communicator furthermore generates a clarifying visual cue to accompany each message. Cues show parts involved in respective operations in the context of workspace layouts. This provides worker support, e.g. for operations that the robot is incapable of and helps communicating planned handover positions. If necessary according to Table 5.1, requests are equipped with buttons to answer either ‘yes’ or ‘no’.

6.1.2. Results

The prototype system offers different software tools that guide users through all stages of system operation. After hardware installation, a graphical user interface supports sensor calibration. With this setup routine completed, tasks can be modelled using the graphical editor described in Section 3.2. Finally, users communicate with the robot system using the smartphone application. The following experiments from the author’s prior work were directed towards validating the hypothesis that end-users can operate this whole toolchain intuitively. Where applicable, the *Questionnaire for the Subjective*

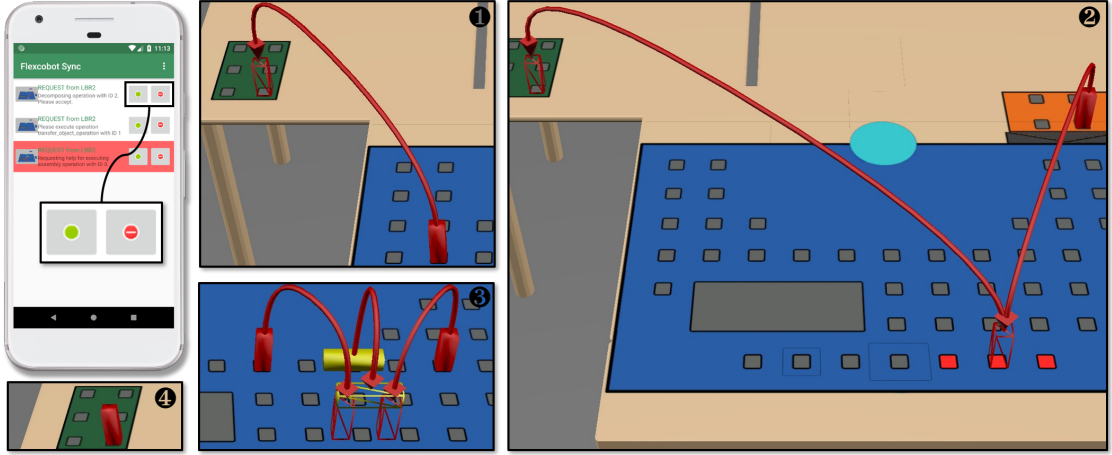


Figure 6.3.: The prototype Communicator implementation presents a list of request-related messages to connected users. Blocking requests that might render the robot incapable of acting if not answered are highlighted (red block). If necessary, two buttons with a red/green icon are generated to answer ‘yes’ or ‘no’. Messages are accompanied by visual cues. The shown image cues (❶-❹) are examples for correspondingly numbered request types in Table 5.1.

Consequences of Use (QUESI) [87] was used to quantify the notion of intuitiveness. This section provides only an abbreviated summary of these user evaluation experiments as the main focus of this thesis lies on an objective, quantitative evaluation of the proposed approach. Detailed results can be found in respective publications.

Initial Setup Phase [140]: A small-scale user evaluation with four participants was conducted to investigate, whether non-expert users can perform the system calibration step successfully within a reasonable amount of time. Against the background of industrial applications in SMEs, students with a technical background were chosen as participants. However, none of them indicated prior concrete experience with similar calibration procedures. They were supplied with a one-sheet user manual describing the calibration user interface (Figure 6.2b). After reading this manual, all subjects managed to calibrate the system appropriately within less than 20 minutes.

Task Modelling Phase [133]: Precedence graphs are used as a task model in this work. They provide the required expressiveness while still being claimed sufficiently comprehensible for end-user programming (Section 2.4). The second user study tried to validate the latter assumption. In a first condition, an overall number of 22 persons participated in the experiment. Users were introduced to precedence graph modelling with an introductory text and a screencast showing how to create a simple model. They were then asked to reproduce the modelling steps shown in the screencast to get to know the procedure and user interface. After that, subjects were instructed to create precedence graphs for three goal states of increasing complexity. The study instructor measured the required modelling times for each goal state and asked the participants to rate intuitiveness with the QUESI questionnaire after completing all task models. Results have shown that the variance of modelling times across participants decreases

steadily for successive task models in the experiment. This can be taken as an indicator for fast learning success. Intuitiveness was rated positively with an above average mean QUESI score of 3.8 on a Likert scale from one to five. This score is similar to those achieved by recent approaches to intuitive robot programming [102, 94]. The second experimental condition involved modelling of graphs with an increasing number of elements, i.e. of involved parts, operation nodes, and precedence relations. Three users with prior experience in operating the editor finished task models with a maximum of 84 elements in less than 10 minutes.

Communication in the Online Teaming Phase [135]: The final component tested with a user evaluation procedure was the smartphone application. To this end, a human subject study was conducted among 18 participants. Each of them was instructed to perform two tasks together with the robot system described in Section 6.1.1. The tasks targeted evaluation of the prototype communication channel and thus involved mostly TYPE 1, TYPE 2 and TYPE 3 operations from the robot point of view. Intuitiveness was rated with a mean QUESI score of 4.1 across all subjects for both tasks. This indicates general acceptance of the approach. However, not all participants were able to complete the tasks successfully – the experiment unveiled clearness and timing of messages as potential sources of issues, particularly in the context of TYPE 3 operations.

6.2. Objective Evaluation

The aim of objective evaluation in this thesis is to explore to what extent production processes can generally profit from flexible human-robot teaming in perspective. To this end, data about numerous teaming processes was gathered with a simulation system and compared to metrics of the fully manual process and optimised human-robot schedules: A set of benchmark tasks that lead to different teaming scenarios (e.g. close proximity working versus independent working on sub-tasks) is introduced in Section 6.2.1. After that, Section 6.2.2 describes the simulation framework that emulates dynamic worker participation in these tasks by modelling partly randomized human decision strategies. This simulation system is parametrised to reflect realistic human pace of work and the discrepancy compared to safe robot operational speeds (Section 6.2.3). Evaluation metrics and reference values to put simulation results into context are outlined in Section 6.2.4. Finally, experimental results are summarised in Section 6.2.5.

6.2.1. Benchmark Tasks

Table 6.1 shows the benchmark tasks that were used in the experiments. Tasks A to D are represented by similarly structured precedence graphs according to Figure 6.4a with a total of 20 operations each. They require shaking, palletising and stacking parts and differ in their initial distribution of objects within the workspace. The part distributions provoke different co-working scenarios: The accumulation of part start and goal positions in Task A forces peers to work in close proximity during the whole teaming process. By contrast, Tasks B and C allow for separation when fetching (B) or delivering (C) resources.

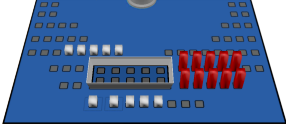
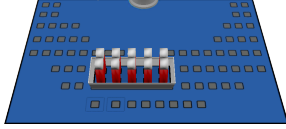
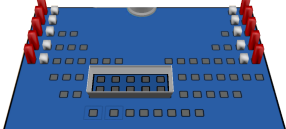
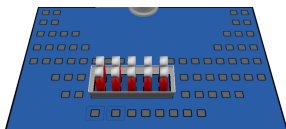
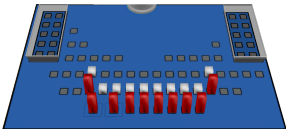
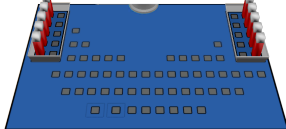
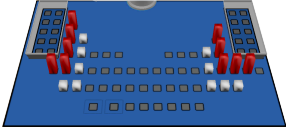
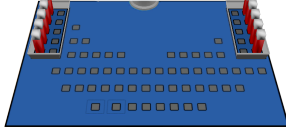
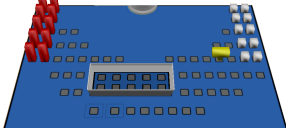
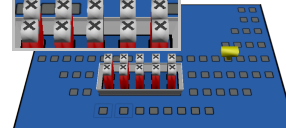
	Initial state	Goal state	Reference data
Task A ($ T = 20$)			$D_H = 96.3 \text{ s}$ $D_R = 181.5 \text{ s}$ $\Sigma_{\text{opt}} = 1.6$
Task B ($ T = 20$)			$D_H = 118.0 \text{ s}$ $D_R = 208.7 \text{ s}$ $\Sigma_{\text{opt}} = 1.6$
Task C ($ T = 20$)			$D_H = 110.9 \text{ s}$ $D_R = 199.8 \text{ s}$ $\Sigma_{\text{opt}} = 1.6$
Task D ($ T = 20$)			$D_H = 88.3 \text{ s}$ $D_R = 190.3 \text{ s}$ $\Sigma_{\text{opt}} = 1.5$
Task E ($ T = 30$)			$D_H = 142.2 \text{ s}$ $D_R = 322.0 \text{ s}$ $\Sigma_{\text{opt}} = \text{n/a}$

Table 6.1.: Benchmark task initial and goal states with reference data for human-only (D_H) duration, robot-only duration (D_R) and optimal speedup (Σ_{opt}) where available (cf. Appendix A)

Finally, Task D requires placing parts in two nearby containers. This can be interpreted as two sub-tasks that can be worked off independently by one agent each. Task E adds synchronisation by sharing a tool. Parts must be stacked and shelved in a container similarly to the aforementioned tasks. The top object additionally needs to be marked by applying the tool. Figure 6.4 shows two alternative graphs that identically lead to the goal state of Task E: Applying the tool to the white parts can be done in parallel to transferring the red ones to the container. The marked objects are stacked onto the red workpieces afterwards (Figure 6.4b). We will refer to this task model as ‘Variant 1’ of Task E. Variant 2 yields less potential for parallel working, as the white parts are marked after stacking them (Figure 6.4c).

The interaction induced by TYPE 2 and TYPE 3 operations depends strongly on human-robot communication. This aspect is better covered by human subject studies and has already been addressed by the results reported in Section 6.1.2. Furthermore, TYPE 1 operations would bias overall task feasibility towards either agent and hence complicate the calculation of comparable performance metrics that solely capture the dynamics of task sharing with the system. This is why all operations of the benchmark tasks are feasible for both human and robot, i.e. $\mathcal{I}_H(\tau) = \mathcal{I}_R(\tau) = \text{TYPE } 0$.

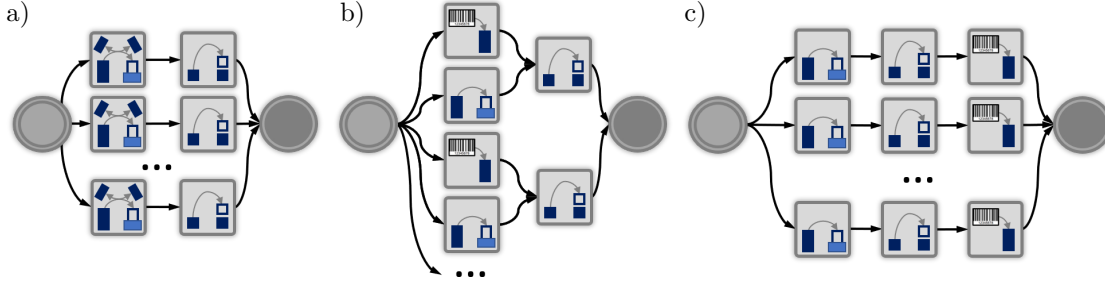


Figure 6.4.: The benchmarks Task A - D have identical precedence graph models that are composed of `PickShake&Place` and `Pick&PlaceOnTop` skills according to Table 3.2 (a). They differ in skill input parameters to achieve different spatial distributions of parts in the workspace. Task E adds tool use with `ApplyLabel` skills. This task can be represented by two different graphs which apply the label before (b) or after placing the white part (c).

6.2.2. Simulation System

The simulation system (Appendix A, Figure A.1) replicates the hardware setup shown in Figure 6.1. A dedicated implementation of the Skill Execution Engine commands a simulation robot controller to move and manipulate parts in the virtual environment. Considering current joint positions of the simulated robot, synthetic point clouds for the Sensor Data Processor are created by rendering the scene from the position where the physical camera would be located.

Objective evaluation targets teaming performance metrics rather than ergonomic aspects. Hence, the simulation of human motion is reduced to movement at constant speed v^H on a roadmap in line with the human workspace model (Section 4.2.1). The current position on the roadmap at time t corresponds to the human presence map H_t , thus simulating human tracking information gathered with the LIDAR sensor in the hardware setup. Against the background of equal partners teaming, it is assumed that humans can transport one part at a time as the robot does. Human part transfers are simulated in four steps: The virtual human is first moved to that point on the roadmap that is least distant to the object in question. Waiting for a constant amount of time $t_{\text{pick}}/2$ emulates the duration of a reaching motion. Then, the part is relocated from the workbench to a part allocation position. Motion of the part while retracting the hand is not simulated explicitly but emulated by waiting for another $t_{\text{pick}}/2$. Placing is achieved analogously by moving to the roadmap point closest to the goal position and considering reaching motions with an overall duration t_{place} .

In addition to interaction with parts, the simulation system needs to make human task allocation decisions repeatedly. In contrast to the robot with its limited view, it is assumed that a worker can fully observe a limited workspace area as shown in Figure 6.1 at any time – decisions are thus always based on full knowledge of operations previously finished by the robot. In doing so, the simulation model accounts for behaviours with different task-related and spatial preferences. The *task related preference* models, whether a human prefers completing subtasks, favours processing parts of a

Human-Aware World Model		
$h_{\text{leg}} = 83 \text{ cm}$	$h_{\text{torso}} = 62 \text{ cm}$	$h_{\text{table}} = 89 \text{ cm}$
$l_{\text{arm}} = 71.5 \text{ cm}$	$\beta_{\text{max}} = 20^\circ$	$\lambda_{\text{crit}} = 33.27 \cdot 10^{-4}$
Simulation System		
$t_{\text{pick}} = 1.30 \text{ s}$	$t_{\text{place}} = 1.44 \text{ s}$	$t_{\text{shake}} = 1.13 \text{ s}$
$v^{\text{H}} = 1.6 \frac{\text{m}}{\text{s}}$	$\omega_{\text{max}}^{\text{R}} = 1.0 \frac{\text{rad}}{\text{s}}$	$\dot{\omega}_{\text{max}}^{\text{R}} = 1.0 \frac{\text{rad}}{\text{s}^2}$

Table 6.2.: Relevant parameter values for experimental evaluation

type before moving on to the next sort of objects or has no preference at all. For the benchmark tasks, subtasks correspond to preferring depth-first graph paths. Parts of a kind are worked off by exploring the task models breadth-first. We will thus refer to the task-related preference manifestations as ‘depth-first’, ‘breadth-first’ and ‘none’. Several active operations may match the task related preference: E.g., a new sub-task must be chosen after finishing a depth-first path, the ‘breadth-first’ strategy involves choosing among possibly several available parts of a type, and the ‘none’ preference allows for all operations that are ACTIVE at a given point in time. The *spatial preference* resolves this ambiguity by preferring the part closest to the human’s current position (‘nearest’) or deciding randomly (‘random’). Random decisions draw an operation with a probability proportional to the point-based part accessibility from the roadmap (Equation 4.11). This way simulated human decisions will likely respect instructions regarding ergonomic occupational safety but may also result in divergent behaviour with a certain probability.

Each pair of a task-related and a spatial preference manifestation corresponds to a human teaming strategy. Strategies involving the ‘nearest’ preference cover deterministic behaviour with workers actively trying to foster parallel working. In Task E for instance ‘depth-first’ combined with ‘nearest’ makes the human simulation issue coherent operations to fill one container before turning to the other one. When seeking to evaluate the overall potential of the approach, considering human participation under these plausible, yet strong assumptions is not sufficient on the one hand – on the other hand, modelling more general human decision-making processes is hardly feasible. The spatial ‘random’ preference accounts for this conflict. By randomisation, we can generate multiple different workflows for some task. This way, the robot coordination algorithms can be tested with a multitude of different decisions that individuals might take so that we can draw conclusions from a statistical evaluation.

6.2.3. Parametrisation

The parameters for simulating cooperative workflows are specified in Table 6.2. They were chosen to represent a realistic shop floor situation – to this end, the body measures for the human-aware world model (Section 4.2.1) were chosen to match the P50 values in ISO/TR 7250-2 [55]. The maximum forward inclination β_{max} follows suggestions of Daub et al. [31] and the workbench height h_{table} was measured in the laboratory setup. These values lead to the derived human reach parameters $r_{\text{arm}} = 44.5 \text{ cm}$ and $r_{\text{max}} = 87.7 \text{ cm}$ for calculating part accessibility (Figure 4.6). World model ageing furthermore relies on the trust factor λ that encodes overall robot trust in stored data (Equation 4.15). Let $\bar{D}_{\text{R}}^{\tau}$

denote an estimate of the average timespan between two robot decisions in consecutive working phases (Algorithm 1). Consider a decision at time t . It is a reasonable strategy to make the system forget any part e until the next decision ($\mathcal{C}_{t+\bar{D}_R}(e) = 0$) if this part was constantly assigned the worst-case human influence term value ($\mathcal{HI} = 1$) during this time period. The trust factor value λ_{crit} in Table 6.2 reflects this strategy for the benchmark tasks and can be used as a starting point to sample the range between instant forgetting ($\lambda = \infty$) and unlimited trust in data ($\lambda = 0$). Further details on the calculation of λ_{crit} are reported in Appendix A. Overall, five trust factor values λ_1 to λ_5 with $\lambda_1 = \infty$, $\lambda_2 = 2 \cdot \lambda_{\text{crit}}$, $\lambda_3 = \lambda_{\text{crit}}$, $\lambda_4 = 0.5 \cdot \lambda_{\text{crit}}$ and $\lambda_5 = 0$ were set for the experiments.

The simulation system relies on parameters to shape the pace of human interaction within the task. Realistic durations for picking (t_{pick}) and placing (t_{place}) parts were estimated by applying the standard motion time system MTM-1 for manual operations in industrial settings [19] (cf. Appendix A for details). The speed of motion on the roadmap of about 1.6 m/s is also derived from MTM-1. The duration t_{shake} of one shaking repetition in the **PickPlace&Shake** skills was set equal to the robot shaking duration. During the simulation experiments the robot moved with a maximum angular velocity $\omega_{\text{max}}^R = 1.0 \frac{\text{rad}}{\text{s}}$ and acceleration $\dot{\omega}_{\text{max}}^R = 1.0 \frac{\text{rad}}{\text{s}^2}$. These values were found acceptable from a human point of view when interacting with the prototype hardware setup, i.e. they are empirical values from laboratory work in the context of this thesis in the first instance. They lie, however, also within the bounds of formal safety regulations as they lead to a maximum cartesian robot TCP velocity of $v_{\text{max}}^R = 0.73\text{m/s}$ in the benchmark tasks (Table A.1): Transient contact between the robot and human arms is assumed to be the predominant safety issue when picking and placing parts in the given setup. For these cases, the v_{max}^R values comply with the speed limits derived in ISO/TS 15066 [56].

6.2.4. Evaluation Metrics and Reference Data

The simulation system collects several performance metrics to quantify the outcomes of teaming processes. These metrics are based on measurements of the following raw data for each run of a task with a certain human strategy and robot system parameter set:

- **Number of successful robot operations** N_{success} : Each robot call to the Skill Execution Engine that leads to successful execution of an operation from the task model increases the number of successful robot operations by one.
- **Number of operation attempts** N_{attempt} : Each attempt to perform an operation from the task model (no matter if successful or failed) is counted by N_{attempt} .
- **Human idle time** D_{idle}^H : Idle time is increased whenever there are no operations that the human peer can do, i.e. whenever there is no active operation that the human is capable of and that has satisfied preconditions.
- **Cooperative task duration** D_{coop} : The cooperative task duration is a measure for the timespan between the moment when a human-robot team started task execution and the point in time when all operation postconditions are satisfied.

The points in time when human and robot become aware of the fact that all postconditions are satisfied may differ: Under the assumption that workers can fully observe the whole workspace, task execution is considered done by the human simulation immediately after the last part has been placed. By contrast, the robot system with its limited view may need additional time to conduct perception operations before the coordination loop terminates. We will therefore distinguish between two task durations as perceived by the human (D_{coop}^H) and by the robot (D_{coop}^R), where usually $D_{\text{coop}}^H < D_{\text{coop}}^R$ holds. Human idle time D_{idle}^H is measured until both agents consider task execution completed.

A reference data set comprising the following quantities was gathered to put the absolute raw metrics into context:

- **Human-only and robot-only task duration D_H and D_R :** The human-only or robot-only duration is measured when either of both agents works, while the other one does not participate in the task at all.
- **Optimal speedup Σ_{opt} :** Let D_{opt} denote the lower bound on task duration that is achieved if both agents are optimally used to capacity. The maximum achievable speedup compared to human-only task duration is given by $\Sigma_{\text{opt}} = D_H/D_{\text{opt}}$ and expresses the cooperative potential of a task.

Human-only durations were gathered by measuring the timespan needed to finish each task with each possible human strategy. Participation of the robot system was disabled during these simulation runs. Robot-only durations were measured accordingly. For randomized strategies, the mean duration of several runs was considered. An adapted implementation of Beumelburg’s capability-based static task allocation approach [16] has provided optimised schedules to determine D_{opt} values that are comparable to D_H durations from the simulation system within the limits of measuring accuracy. Reference values for each task are listed in Table 6.1. Refer to Appendix A for more detailed information on reference data acquisition.

The below derived metrics can be determined based on reference values and raw data:

- **Robot participation rate Π_R :** For a task model $(T, <_T)$ with $|T|$ operations, the participation rate is defined by $\Pi_R = N_{\text{success}}/|T|$. This score captures the fraction of the task that the robot has handled.
- **Robot error rate \mathcal{E}_R :** The error rate is given by $\mathcal{E}_R = 1 - N_{\text{success}}/N_{\text{attempt}}$. It is a measure for robot decision quality and in turn for interference between human and robot decisions.
- **Cooperative speedup Σ_{coop} :** The cooperative speedup relates cooperative task duration D_{coop} to the time a task would take a human agent alone. It is calculated according to $\Sigma_{\text{coop}} = D_H/D_{\text{coop}}$ and expresses the acceleration that is reached by the investigated flexible teaming method compared to fully manual work.

In analogy to cooperative task duration, we will consider two versions Σ_{coop}^H and Σ_{coop}^R of speedup that are calculated by using either D_{coop}^H or D_{coop}^R .

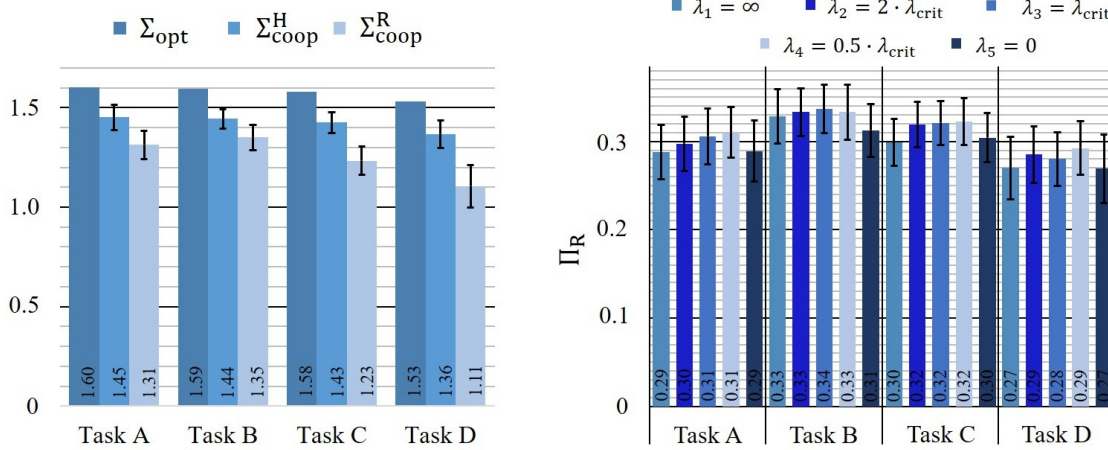


Figure 6.5.: The relation of optimal speedup (Σ_{opt}) and mean speedup values achieved with the proposed system (Σ_{coop}^H and Σ_{coop}^R) suggest an overall potential to accelerate tasks (left). System performance in terms of the robot participation rate (Π_R) varies slightly with robot trust in data which is encoded by the trust factor value λ (error bars indicate standard deviations).

6.2.5. Results

The evaluation results presented hereinafter were gathered with the following procedure: The robot system was tested with each human strategy supported by the simulation system for each of the tasks in Figure 6.4, i.e. the approach was tested with each human preference pair in $\{\text{'depth-first'}, \text{'breadth-first'}, \text{'none'}\} \times \{\text{'nearest'}, \text{'random'}\}$. This procedure was repeated twice for most task models: At first, the robot system worked without knowledge about human positions in the workspace by activating the set of interaction indicators $F_1^{\mathcal{HI}}$ for world model ageing (cf. Section 4.2.3). Additional sensor data from LIDAR scans were used in a second experiment with $F_2^{\mathcal{HI}}$. Each interaction indicator set was moreover paired with each of the trust factor values λ_1 to λ_5 as derived in Section 6.2.3. An additional test set where the robot system made completely random decisions rather than relying on the human-aware world model in Algorithm 2, Line 3 was considered to verify the overall usefulness of the data ageing heuristics. Appendix A presents a detailed listing of the experiment protocol - the evaluation data set is based on a total of 3240 simulated teaming workflows, each reflecting different decisions of both human and robot.

Overall System Performance

Figure 6.5 (left) compares the maximum achievable, optimal speedup Σ_{opt} to the speedup values that result from flexible teaming. World model data ageing with knowledge about human presence was enabled for this experiment (interaction indicator set $F_2^{\mathcal{HI}}$). The values reported for Σ_{coop}^H and Σ_{coop}^R are mean values with their standard deviations across all simulation runs with different human strategies and trust factors per task, i.e. each value summarises 300 different workflows (cf. Appendix A). The proposed system

makes heuristic and therefore possibly erroneous decisions as a consequence of partial workspace observability. Aborted skill execution attempts can be time-consuming and lower the attainable speedup as well as motion for condition evaluation that does not lead to confirmed part availability and, subsequently, successful operations. Still, we can observe that task duration is significantly shortened by about 30% when looking at the speedup Σ_{coop}^H . This value is plausible when taking the difference in human and robot pace of work into consideration – even load of both agents cannot be expected under the given parametrisation (Section 6.2.3). Findings regarding the speedup are furthermore in line with measurements of the robot participation rate Π_R . This rate also amounts to around 30% depending on the concrete task and trust factor value (Figure 6.5, right). The relation between optimal speedup Σ_{opt} and actual speedup Σ_{coop}^H is consistent across all tasks – about 90% of the optimum are achieved.

The Σ_{coop}^H values are based on the D_{coop}^H metric which stops counting task duration in the moment when all postconditions are satisfied. The robot system is, however, only done with a task and ready for the next job after the duration D_{coop}^R . Technically speaking, the speedup Σ_{coop}^R would therefore be the decisive metric. According to this metric teamwork leads to an approximate acceleration of 23% for Tasks A to C and of 10% in Task D. This substantial decrease compared to Σ_{coop}^H identifies the final phase of task execution as a weakness of the current system realisation: The difference $D_{\text{coop}}^R - D_{\text{coop}}^H$ in durations from the human versus the robot point of view emerges from the additional time that the robot needs to observe that all operations have been finished. This step can, however, be eliminated e.g. by implementing a simple system feature that allows humans to stop the robot – Σ_{coop}^H is therefore considered the better suited metric to judge the overall potential and future prospects of the flexible teaming concept.

Influence of Robot Trust in World Model Data

Robot trust in world model data is controlled by the trust factor value λ that is used for world model ageing. The mean robot participation rate Π_R in each task is broken down by λ in Figure 6.5 (right). The absolute difference between the individual modes of forgetting instantly (λ_0), incrementally lowering the pace of forgetting (λ_1 to λ_4) and never losing trust in parts once they were sensed (λ_5) is low – hence, the data set cannot suggest a unified value of λ that is optimal across all tasks. Yet there is a trend towards modes where certainty decreases gradually over time: Participation rates achieved with λ_2 , λ_3 and λ_4 consistently lie above those for λ_0 and λ_5 in all tasks. Ageing with at least one of these λ values provides an advantage of about 2% for each task when compared to instant forgetting and unlimited trust. This indicates that taking the human-aware world model as a foundation for task allocation is conducive to teaming efficiency.

Table 6.3 lists mean robot error rates \mathcal{E}_R and human idle times D_{idle}^H with their standard deviations for different ageing strategies. Skill execution attempts are always preceded by an evaluation of preconditions if relevant parts are only slightly aged (cf. Figure 5.6 and Section 5.2). This means that the error rates for λ_1 , λ_2 , λ_3 and λ_4 express direct interference among agents in situations when the human picks a part that the robot has

		$\lambda_1 = \infty$	$\lambda_2 = 2 \cdot \lambda_{\text{crit}}$	$\lambda_3 = \lambda_{\text{crit}}$	$\lambda_4 = 0.5 \cdot \lambda_{\text{crit}}$	$\lambda_5 = 0$
Task A	\mathcal{E}_R [%]	6.38 ± 9.76	8.81 ± 11.24	6.64 ± 9.26	7.75 ± 10.08	42.12 ± 11.52
	D_{idle}^H [s]	6.62 ± 3.69	7.52 ± 4.4	8.09 ± 3.13	8.13 ± 3.98	11.76 ± 6.12
Task B	\mathcal{E}_R [%]	4.30 ± 7.81	6.41 ± 9.29	7.45 ± 8.87	6.59 ± 8.93	37.24 ± 11.33
	D_{idle}^H [s]	6.28 ± 4.07	5.66 ± 3.89	7.67 ± 4.94	7.04 ± 4.66	10.69 ± 5.65
Task C	\mathcal{E}_R [%]	8.00 ± 10.54	8.01 ± 8.8	6.93 ± 8.45	6.19 ± 9.48	37.29 ± 11.7
	D_{idle}^H [s]	14.47 ± 7.25	13.43 ± 6.9	12.70 ± 6.77	13.75 ± 7.13	17.39 ± 6.55
Task D	\mathcal{E}_R [%]	6.14 ± 9.56	7.25 ± 9.72	6.71 ± 9.21	6.47 ± 8.30	50.52 ± 8.25
	D_{idle}^H [s]	15.50 ± 8.23	15.78 ± 8.05	15.26 ± 8.9	14.72 ± 7.26	25.78 ± 9.15

Table 6.3.: Mean robot error rates \mathcal{E}_R and human idle times D_{idle}^H with their standard deviations across human strategies for different tasks depending on the patience constant λ

just started to approach. The mean values are overall low but scatter strongly. This suggests a dependency of system interplay with different human strategies and individual decisions. The error rate for unlimited trust (λ_5) is significantly higher. Since this world model ageing mode keeps all parts at a certainty value of 1 the current system implementation will never check preconditions actively – robot operations are thus often triggered haphazardly for $\lambda = 0$. This does not show up as an overly strong decrease in robot participation rates in Figure 6.5 (right) due to the rather low duration of a failed pick action – it should still be avoided by preferring $\lambda > 0$ as frequent interference might impact user experience negatively.

Each task shows a level of human idle time that is similar across the preferred trust factor values λ_1 to λ_4 . The tasks considered in this experiment do not involve tool sharing – idle times can thus only emerge in two cases at the end of task execution: (i) The human agent may need to wait for the robot to place the last red part to put the white one onto it. (ii) Waiting for the robot to detect that all operations are done may be necessary. Observations of simulation processes identify the latter situation to be the primary impact factor on idle time. We can thus say that $D_{\text{idle}}^H \approx D_{\text{coop}}^R - D_{\text{coop}}^H$ for the considered pick-and-place Tasks A - D. This waiting time can easily be avoided as discussed in the context of overall system performance. Human idle times are thus overall low and can be reduced further to the amount of few seconds.

Influence of Sensor Data

A major goal of this work is to reduce the number of sensors used by the system to attain a lean and achievable robot system. The next experiment investigates whether the additional LIDAR sensor contributes to teaming performance positively. To this end, Figure 6.6 compares average speedup values (left) and robot participation rates (right) for simulation runs with the interaction indicator sets $F_1^{\mathcal{H}\mathcal{I}}$ (without LIDAR data) and $F_2^{\mathcal{H}\mathcal{I}}$ (with LIDAR data). The results are based on world model ageing with $\lambda = \lambda_{\text{crit}}$. Values are furthermore compared to data that results when the robot makes random decisions rather than referring to the world model with its heuristic certainty values.

The participation rate indicates a positive influence of world model ageing with LIDAR

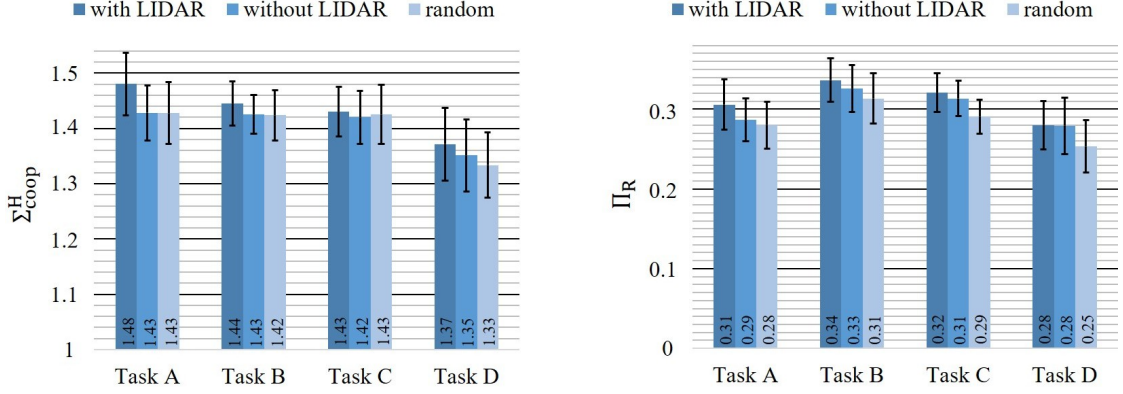


Figure 6.6.: Simulations with $\lambda = \lambda_{crit}$ show that mean participation rates (Π_R , with error bars indicating standard deviations) increase when comparing decision making without world model ageing (random), with world model ageing based on a human workspace model (without LIDAR data) and ageing with precise LIDAR data on human presence (with LIDAR data). Differences in speedup (Σ_{coop}^H) are comparatively low due to the dominant human influence on task duration.

data. Results for Π_R in this mode lie above those attained with decisions based on the interaction indicator set F_1^{HI} across all tasks. Random decisions yield the lowest Π_R values with a consistent difference of about 3% in comparison to the best achieved rates. This substantiates the use of world model ageing for handling partial observability. The equal levels of Π_R for F_1^{HI} and F_2^{HI} in Task D are plausible as this task allows for the most spatial distancing of agents. This reduces the importance of spatially differentiated data ageing in contrast to more narrow scenarios. Speedup values for Task A corroborate this assumption – the maximum observed difference in speedup when comparing ageing with and without LIDAR data was measured for this narrow workspace setup.

The absolute difference in speedup when varying sensory input for robot decision-making is rather small. This is attributable to the realistically large difference in human and robot pace of work as particularly expressed by the duration of tasks when performed by either agent alone (cf. D_H and D_R in Table 6.1). This discrepancy renders human decisions the dominant impact on teaming performance and reduces the effect of changes to the system on measurements under the chosen parametrisation. One would still expect speedup measurements that are similarly graduated as participation rates – this cannot consistently be observed for the randomised robot decision strategy in Figure 6.6. A possible reason is that human and robot may work against each other in certain constellations of task models and human strategies: The robot can pick parts that the human simulation has started to approach just as the human simulation can take an object away during the robot approach motion. This may lead to human detours and decreased productivity with the aforementioned strong impact on teaming performance. In turn, lower participation rates due to random decisions might produce less interference and ultimately better performance in such situations. The measured metrics were not designed to capture these aspects – further experiments are needed to clarify this effect.

Influence of Human Strategies and Task Model Structure

The strong scatter of mean robot error rates in Table 6.3 has already indicated a dependency between system performance and individual human strategies of the simulation system. Figure 6.7 breaks the results of Figure 6.6 (left) down into charts that further clarify this influence for each task. Selecting operations with parts that are probably available according to their certainty in the human-aware world model is intended to make the robot avoid areas where its partner is currently working. The best speedup values were measured for combinations of human strategies with the spatial ‘nearest’ preference and decisions incorporating LIDAR data across all tasks. The ‘nearest’ preference makes the human work locally and therefore complements the robot decision-making scheme when using $F_2^{\mathcal{HI}}$. The data thus suggests that the intended spatial separation is generally achieved. The advantage of decisions with the interaction indicator that uses LIDAR data for world model ageing is, however, not given for all human strategies. Observations of simulation runs show in particular that the system cannot adapt fast enough if the human agent alternates between different far apart locations quickly. We can furthermore observe by the example of Task C that random robot decisions may sometimes even yield the best performance. Aside from the fact that these results are still subject to statistical effects this points out that the heuristic robot strategy is based on limited sensory input and can hence not adapt to human decisions in every case. This section is intended to investigate the future prospects of flexible teaming in general by considering a cross section of all feasible workflows to complete a task. Individual unfavourable cases are an essential part of this cross section acquired by partial randomisation of decisions – their in-depth analysis thus exceeds the scope of this generalised assessment.

Significant differences in performance occur when considering the two possible precedence graph structures for Task E (Figure 6.4). Figure 6.8 puts the speedup and participation rates for $\lambda = \lambda_{\text{crit}}$ with either task model into relation. The shown charts enable two major observations:

- Considering the mean achieved speedup across human simulation strategies shows that Variant 1 of the task model leads to overall better results.
- For a given human strategy, the robot participation rate can drop when choosing an unsuitable task model.

For instance, the system will co-work more productively with a human who follows strategy E if Variant 1 is chosen. By contrast, Variant 2 is matched better by human preferences according to strategy F. The choice of a concrete precedence graph model for some task can thus have a considerable impact on teaming performance when working with the proposed system.

This is also evident from human idle time measurements: Table 6.4 reports the percentage $\hat{D}_{\text{idle}}^{\text{H}}$ of human idle time that is caused by delays in resource allocation as measured in simulated workflows with $\lambda = \lambda_{\text{crit}}$. This metric is calculated by subtracting the influence of idle time $|D_{\text{coop}}^{\text{R}} - D_{\text{coop}}^{\text{H}}|$ due to robot perception at the end of a task and relating the

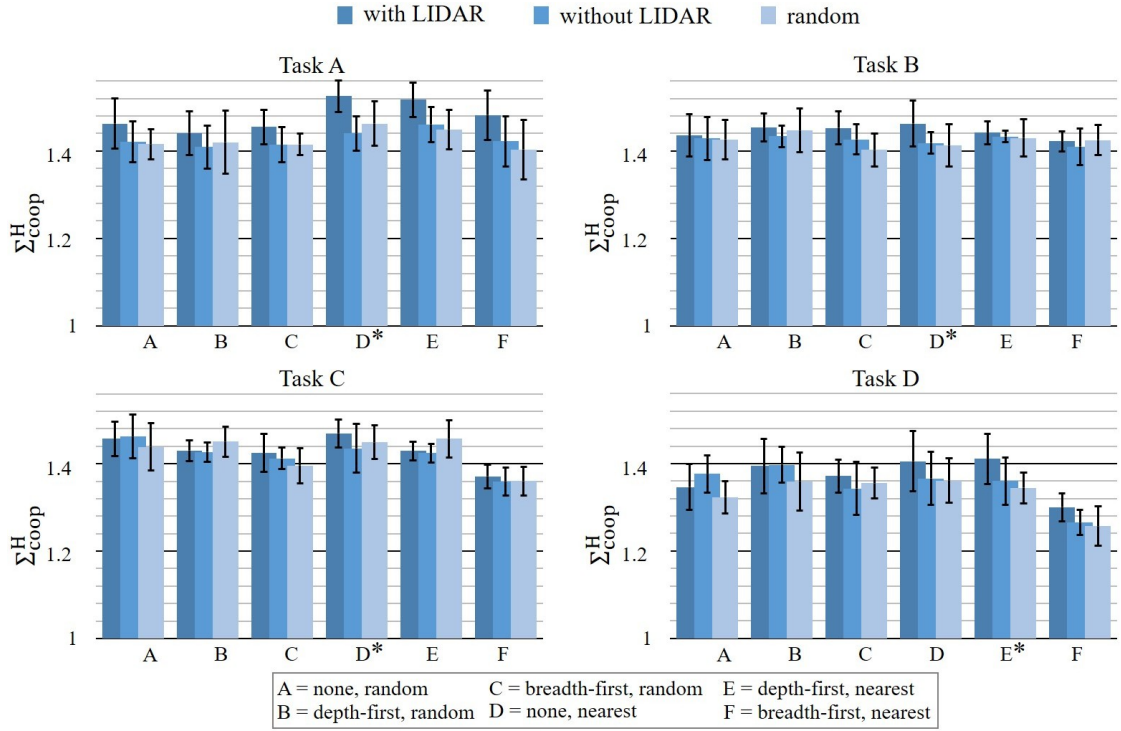


Figure 6.7.: Speedup mean values and standard deviations vary depending on the task model and human strategy when changing the level of available sensor data for robot task allocation decisions. The human strategy yielding the highest speedup value for each task is marked with *.

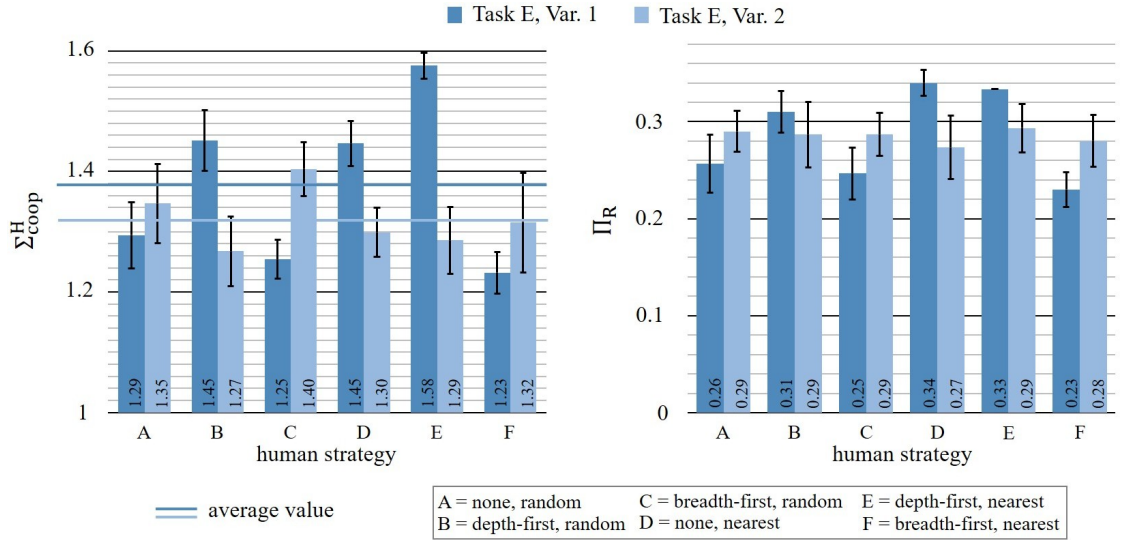


Figure 6.8.: Different precedence graph models for the same task start and goal state yield varying teaming performance (mean speedup Σ_{coop}^H and Π_R with standard deviations) depending on the human working strategy.

	$\hat{D}_{\text{idle}}^{\text{H}}$ [s]					
	without tool sharing				with tool sharing	
	Task A	Task B	Task C	Task D	Task E Var. 1	Task E Var. 2
none, random	0.5 ± 1.3	1.9 ± 3.7	2.2 ± 3.7	2.1 ± 3.9	4.4 ± 6.5	8.1 ± 6.9
depth-first, random	3.6 ± 3.5	4.7 ± 5.6	4.2 ± 5.4	1.9 ± 3.8	5.7 ± 4.9	13.5 ± 5.5
breadth-first, random	0.9 ± 1.5	0.7 ± 1.6	1.3 ± 2.1	0.3 ± 1.0	6.7 ± 8.9	1.1 ± 1.8
none, nearest	1.0 ± 1.8	0.5 ± 1.0	1.8 ± 1.7	0.2 ± 0.6	3.6 ± 4.6	9.3 ± 6.5
depth-first, nearest	2.3 ± 3.2	6.0 ± 4.4	0.9 ± 1.8	2.4 ± 3.4	0.9 ± 2.7	15.3 ± 3.7
breadth-first, nearest	0.5 ± 1.6	1.7 ± 2.7	1.6 ± 2.3	0.0 ± 0.0	8.8 ± 5.4	2.9 ± 3.7

Table 6.4.: Human idle time percentage $\hat{D}_{\text{idle}}^{\text{H}}$ (with standard deviation across simulation runs) resulting from waiting for resource availability

	\mathcal{E}_{R} [%]				
	$\lambda_1 = \infty$	$\lambda_2 = 2 \cdot \lambda_{\text{crit}}$	$\lambda_3 = \lambda_{\text{crit}}$	$\lambda_4 = 0.5 \cdot \lambda_{\text{crit}}$	$\lambda_5 = 0$
Task E Var. 1	17.00 ± 12.01	17.48 ± 12.23	18.29 ± 11.93	17.06 ± 10.37	43.62 ± 10.97
Task E Var. 2	14.92 ± 10.72	14.02 ± 10.11	13.91 ± 9.03	12.82 ± 8.76	42.64 ± 11.22

Table 6.5.: Mean robot error rates \mathcal{E}_{R} and their standard deviations across simulation runs of Task E with shared access to a tool

remaining idle time to the duration $D_{\text{coop}}^{\text{H}}$ as perceived by the human, i.e.

$$\hat{D}_{\text{idle}}^{\text{H}} = \frac{D_{\text{idle}}^{\text{H}} - |D_{\text{coop}}^{\text{R}} - D_{\text{coop}}^{\text{H}}|}{D_{\text{coop}}^{\text{H}}}. \quad (6.1)$$

This fraction lies within a range of few percent points for Tasks A to D. As discussed in the context of Table 6.3, $\hat{D}_{\text{idle}}^{\text{H}}$ can here only result from seldom situations at the end of task execution. In comparison, the values for both variants of Task E show a slight increase of $\hat{D}_{\text{idle}}^{\text{H}}$ that can be traced back to the fact that both agents share a single tool for `ApplyLabel` operations. Respective values are mostly not remarkably higher than those for the tasks without tool sharing. We can, however, observe again that each human strategy matches better with either Variant 1 or 2 of Task E. Hence, idle times differ strongly depending on the task model. The values for strategies involving the ‘depth-first’ preference stand out in particular. As also observed from the corresponding participation rates (Figure 6.8), unfavourable design of the precedence graph may lead to decreased performance – in the case of Task E, this can manifest in workers waiting for the robot to release the tool for up to 15.3% of the overall task execution time.

Influence of Tool Sharing

Task E differs mainly from the other benchmark tasks due to the shared access of both agents to a single tool. We have already seen from the data in Table 6.4 that this leads to a moderately increased level of human idle times in most cases when compared to Tasks A - D. The impact of shared resources on robot error rates is more significant. Respective mean values in Table 6.5 show that \mathcal{E}_{R} takes more than double the value reported for all other tasks (Table 6.3).

6.3. Conclusions

Summary

This chapter has first described a laboratory prototype that implements the proposed flexible teaming concept (Section 6.1.1). The prototype relies on few low-cost sensors. It integrates with the task modelling approach by use of printed workspace layouts to partly structure the workbench and thus simplify non-expert user studies. The system supports picking and placing, basic part processing and assembly. Explicit communication is achieved by a smartphone messaging app. The robot sends messages to workers via this app. Humans can in turn accept or reject requests by tapping corresponding buttons. The smartphone application is part of a software toolchain for end-user operation of the overall system: In addition to the graphical task model editor (Section 3.2.1) the toolchain also provides a user interface to support the calibration process needed for the RGB-D camera and the LIDAR sensor used by the system. Subjective evaluation results were gathered by conducting user studies with individual parts of the prototype toolchain (Section 6.1.2). Results suggest that users can handle sensor calibration and modelling of a complex precedence graph task model in about 30 min. Communication with the smartphone app was also found intuitive by human subjects in the study.

The focus of this thesis lies on the technical aspects and potential of a concept for flexible, dynamic teaming. Objective evaluation metrics to quantify system performance were therefore gathered with a simulation system (Section 6.2.2). This system is capable of emulating worker decisions with different plausible strategies and preferences (e.g. working spatially local or preferring to complete sub-tasks). Individual strategies incorporate a varying degree of randomization in human decisions to account for the fact that human decision-making processes cannot fully be modelled. Hence, the simulation system is suited to acquire statistical measures of teaming performance for differently shaped tasks, with different robot world model ageing strategies etc. To this end, a set of benchmark tasks was designed (Section 6.2.1). Respective tasks use different skills and vary the spatial distribution of parts in the workspace to support different teaming scenarios (e.g. close proximity working or independent sub-tasks that can be done in spatial separation). More than 3000 teaming workflows were simulated – in doing so, the simulation system was parametrised to reproduce a realistic discrepancy between the human and robot working pace. The major finding of these investigations is that the benchmark tasks could be accelerated by about 25 - 30% on average compared to fully manual work. This means that competitive speedup values could be achieved in comparison to static teaming with optimised schedules despite the heuristic robot decision-making scheme. A comparison with purely random robot decisions has confirmed that the proposed human-aware world model is beneficial for task allocation decisions under partial observability. This comparison has moreover shown the necessity of precise human positional data – world model ageing solely relying on a static human model yielded less robot participation in tasks. All in all, flexibility and a lean hardware setup in terms of minimised sensor use can thus be said to still offer the prospect of productive partial automation within the boundary conditions of the conducted experiments.

Discussion

Experiments with the simulation system have unveiled that the robot can require a notable amount of time to perceive that a task has been finished after the last part has been placed. The full speedup as stated above can only be reached when this time span is not considered as human idle time – respective values thus represent a theoretical prospective measure for the overall potential of the approach. Using the system in its current state can still reduce task durations by about 10 - 25% depending on the task model while achieving overall low human idle times and robot error rates. The full potential can be exploited after minor changes to the system implementation, e.g. by adding a feature that enables users to signal task completion to the robot to shorten the final perception phase. A more sophisticated perception planning method to enhance the information gain during any robot motion may generally accelerate progress detection in the course of task execution.

The actual performance for a given task has been observed to depend on the human strategy. The robot system seeks to establish spatial separation from the current human working area and cannot always follow a partner that alternates between different locations quickly – the capability of the system to adapt to different working styles is hence bounded. This effect has particularly been observed for tasks that can be represented by different precedence graphs: Significant human idle times caused by delayed access to shared tools can arise and reduce system performance if an unfavourable task model is chosen. This needs to be considered when creating precedence graphs. However, compatibility of task models, human strategies and robot system behaviour is a complex question that cannot be expected to be answered by end-users. The experiments thus raise new requirements regarding the proposed task modelling approach: Users may need additional guidance to create suitable graphs when working with the graphical task model editor. This topic has not been taken into account so far. Furthermore, using a tool that is shared by both agents leads to moderately increased human idle times, but also to notable rates of aborted robot operation attempts. This can be traced back to direct interference of agent actions when the human claims the tool while the robot has also started to grasp it. Although not impacting robot participation rates too strongly this is an undesirable effect with a potentially negative impact on user experience.

All aforementioned limitations can be attributed to the imperfectness of decision-making under partial workspace observability. Limited knowledge about the world state and a partner's recent actions must, of course, complicate mutual adaption and therefore impact teaming performance negatively – this has even lead to view constellations in which purely random robot decisions have outperformed the proposed heuristic approach in the experiments. Still, the results suggest that a cost-efficient setup with view sensors can lead to reasonable gains in productivity by human-robot teaming on average. The results must, however, be interpreted in consideration of the assumptions that were made to simulate human participation: The partly randomized simulation does not necessarily match plausible human motion and decisions in any case. It was moreover assumed that workers transfer only one object at a time to create a basic level of equal partnership with a single-arm manipulator. Future work should therefore seek to relax this assumption and

address the question of scalability by pairing a more sophisticated two-handed human model with a comparable dual-arm manipulator.

A need for further investigations arises also from the human subject studies: Issues regarding clearness and timing of messages sent by the robot confirm the prior assumption (Section 5.4) that further work with regard to the human cognitive workload is necessary. Furthermore, individual toolchain components have so far only been evaluated in isolated user studies – a comprehensive study, where particularly the teaming process is evaluated with a stronger focus on the effects of flexibility when needing to leave the workbench temporarily is a potential next topic that can be addressed based on the experiences gathered with the simulation system.

CHAPTER 7

Conclusions

7.1. Summary and Discussion	115
7.2. Future Work	119

WITHIN this final chapter, Section 7.1 reviews the proposed approach, the experimental findings and contributions of this thesis. To conclude with, recommended directions for future work are outlined in Section 7.2.

7.1. Summary and Discussion

The working hypothesis of this thesis was that flexible, dynamic task sharing under partial workspace observability can enable beneficial human-robot teaming based on shared task models created by domain experts. In particular, the key goal was to enable a mode of co-working that supports a mixture of human-robot coexistence, cooperation and close proximity collaboration. Against the background of applications in small and medium-sized enterprises, this was to be achieved by a concept that supports the existing workforce in operating the system by not relying on expert knowledge on robotics. In addition, sensor use was bounded as far as possible to reduce the necessary initial investment and costs for setting up the workplace. The following answers to the research questions posed in Section 1.3 generally confirm the working hypothesis – yet they also point out important limitations that suggest starting points for future investigations:

Q1 To what extent can end-users without knowledge on robotics use a graphical interface to create task models that can be shared with robots for flexible teaming?

A review of task models used in related work has shown that precedence graphs are a reasonable candidate for representing tasks to be shared by flexible teams (Section 2.4). They are widely used, maintain maximal parallelism and can be claimed sufficiently comprehensible for non-expert users. Chapter 3 has introduced a graphical task modelling tool to validate the latter assumption by applying insights from prior art on intuitive CAD-based and icon-based robot programming: A three-layered skill framework introduces

levels of abstraction between hardware control commands and process steps on a human-legible task level. Skills are internally defined by a graph structure that enables automatic derivation of operation pre- and postconditions (Section 3.1). These conditions are necessary for guiding robot perception of task progress. They are furthermore a basis for visualizing parts and the effects of skills in a virtual representation of the shared workspace. Concrete process steps are instantiated by choosing a skill as a template. Input parameters are then specified by selecting parts and locations in the virtual workspace. Fully parametrised skills are represented by an icon each – users are queried to connect these icons with graph edges to specify the temporal dependencies of a precedence graph structure (Section 3.2). A user evaluation of the task editor suggests that non-experts can quickly learn to share procedural task knowledge with the robot system (Section 6.1.2). The approach was overall found intuitive. Given a basic level of experience with the editor, users were even able to construct task models with about 80 graph elements in less than ten minutes. In summary, these results indicate that ideas from intuitive robot programming can well be adapted to enable users without robotics expertise to set up complex task models for flexible teaming. However, a limitation of this approach lies in the ambiguity of precedence graphs: A single graph does not necessarily cover all possibilities to achieve a goal, i.e. there may be two or more different task structures leading to the same goal state (Figure 6.4). Simulation experiments have shown that an unfavourable choice among these task model options can impact teaming performance negatively. Further investigations are thus required towards supporting users in generating models that are advantageous with regard to their working habits.

Q2 How far can a robot system make meaningful dynamic task allocation decisions under partial workspace observability without extensive prior training?

Robot task allocation decisions are the key to dynamic task sharing. Accepting partial workspace observability in exchange for a convenient to set up, affordable hardware configuration with view sensors renders these decisions challenging: Parts that the robot has previously detected with a camera mounted near the robot hand can be modified while they are out of the sensor field of view. Decisions are therefore made on possibly outdated and hence partial knowledge of the world state and task progress. A human-aware world model was introduced in Chapter 4 to enable judging the reliability of previously sensed data. To this end, a certainty measure is calculated incrementally over time for each stored part (Section 4.2). This measure decreases proportionally to a heuristic estimate of the likelihood of parts being modified by human agents. The potential for workers taking influence on parts is estimated in consideration of spatial part accessibility and relevance to upcoming operations in the task (Section 4.2.2). This heuristic solely relies on a human handling area model inspired by ergonomic considerations, the task model and a task progress estimate deduced from observed parts matching operation postconditions. Additional sensor data from a laser range finder was used to investigate the impact of human positional data while only slightly complementing the set of required sensors. A small-scale user evaluation suggests that necessary calibration routines for these sensors can even be performed by non-experts within a reasonable timespan (Section 6.1.2).

This supports the aim of system operation by the existing workforce in SMEs. Based on the world model, just-in-time robot decisions can be made in favour of operations that involve parts with high certainty values – these parts are likely still available at the stored locations (Sections 4.3 and 5.3.1). This strategy was evaluated in combination with different, partly randomized human working habits in a simulation system to investigate performance in multiple workflows resulting from dynamic decisions (Section 6.2.5). A low fraction of aborted skill execution attempts indicates little interference with human decisions. However, access to a shared tool has been shown to increase human idle times and robot error rates. The limitations of decisions under partial observability are pointed out by few constellations of tasks and human strategies in which random robot decisions yielded better performance measures than those emerging from the heuristic task allocation scheme. By construction, the system prefers to work in areas that are currently not occupied by humans to foster efficient parallel working. The approach therefore works best with complementary human habits that lead to spatially local working without interference – prior training or incremental learning of typical human ways of working for individual tasks could therefore support increased robot adaption. Still, performance measurements with the approach as is are overall promising: The results show that the proposed method enables the robot to handle about 30% of the operations within benchmark pick-and-place tasks on average. Taking typical discrepancies between human and robot pace of work due to safety regulations into account, this is a reasonable robot contribution to task completion.

Q3 How can dynamic task allocation be combined with flexible communication and a team mental model to integrate coexistence, cooperation and collaboration?

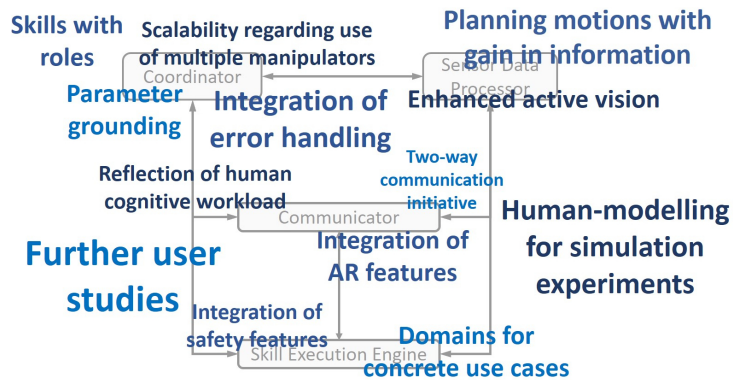
Having taken a task allocation decision, the robot system needs a strategy to coordinate this decision with its partners. For instance, an operation can be handled by checking preconditions and issuing skill execution afterwards. Aborted skill execution indicates that some other agent must have claimed necessary resources in the meantime. In this situation, monitoring of the operation postconditions may unveil task progress achieved by teammates. If progress cannot yet be confirmed this way, the required parts might have been returned to their expected locations. It is then reasonable to retry the operation later by re-iterating the aforementioned steps. This strategy presumes that parts can always be accessed by the robot – the task modelling approach does, however, not constrain possible locations of parts in the workspace to areas that the robot can reach. It rather considers the tabletops of all available workbenches on the shop floor. This leads to regions that are exclusively accessible to either humans or the robot. Operations that are feasible for both agents alike, for only one of them or even for neither of them individually in the case of collaborative process steps can thus well be part of a task model (Section 5.1.1). This work has therefore proposed to formulate individual, exchangeable strategies for different agent capability constellations: Operations are first grouped into categories according to their feasibility for individual agents and the resulting need for interaction (Section 5.1.2). A state machine is used to approximate a team mental model for each of these interaction categories. These component mental models encode the

robot system's understanding of the necessary course of actions for handling operations of respective categories (Section 5.1.4). To this end, states are designed to trigger skill execution, active perception of operation conditions, or human-robot communication. Following transitions in the state machine assigned to some operation thus produces an expedient robot strategy towards advancing this operation. The proposed system can hence particularly (i) try to execute operations as outlined above by itself if possible, (ii) notify humans about operations that they must take care of (iii) and communicate to establish mutual commitment before engaging into collaborative operations. The state machines are preemptive in a way that control is passed back to the superordinate dynamic task allocation procedure after as few state transitions as possible – this ensures reactivity to keep the system capable of acting rather than pursuing a previous, potentially unfavourable decision at all costs (Section 5.3). Frequent re-planning is especially supported by a flexible mode of communication: Messages are predominantly exchanged in an asynchronous fashion, thus preventing robot productivity to come to a standstill if partners are currently otherwise occupied and therefore do not answer requests (Section 5.1.3). The prototype system implements this mode of communication by exchanging messages via a smartphone application (Section 6.1.1). All in all, maintaining the capability of acting at all times leads to a level of robot decision authority and system autonomy that enables independent parallel working in coexistence as well as cooperation. Different strategies to process operations depending on agent capabilities furthermore integrate communication that is particularly necessary for entering collaboration – the technical foundations of flexible teaming under partial workspace observability were thereby laid by an extensible framework. The interaction scheme was generally rated positively in a preliminary evaluation procedure with human subjects (Section 6.1.2). Yet, the amount and timing of messages as sent by the current implementation hampered intuitive use. In addition, the initiative to initiate communication is currently biased towards the robot agent. Hence, adjusting the component team mental models and improving the communication protocols to achieve more elaborate two-way communication while putting emphasis on humans' cognitive workload are necessary next steps to advance the proposed approach. This work was moreover conducted based on the assumption of skilled, cooperative workers who will not introduce faults into the process – this shortcoming can be overcome by extending the system architecture towards error handling as outlined in Section 7.2.

Q4 To what extent is the proposed method beneficial for shared task execution? How does the flexible approach compare to static methods for task sharing?

Let us finally look at the overall future prospects of applying flexible human-robot teaming based on dynamic task sharing with a limited sensor setup. A major finding of the simulation experiments was that task execution could be accelerated, depending on the concrete task structure and human strategy, by at least 10 - 25% with the current system implementation. In perspective, execution times may even be reduced by a total of 25 - 30% on average with only minor changes to the way how the end of a task is detected (Section 6.2.5). Respective experiments were conducted with tasks taken from

Figure 7.1.: Several proposed directions of future work can be associated with the architectural components of the system.



a benchmark domain with actions that are also usually used in collaborative studies with applications to assembly and manufacturing (Section 3.1.3). The proposed approach can therefore be said to be beneficial in that it achieves a measurable gain in productivity for synthetic tasks that relate to typical scenarios in SMEs. Comparing the results to one of the typically used optimisation approaches for generating optimal schedules, flexible teaming has been shown to achieve competitive speedups despite the limited use of sensors. The data must, of course, be interpreted in consideration of the abstract use case and the limitations of modelling human participation in the simulation experiments – usefulness for concrete tasks must still be shown. Nevertheless, this thesis has contributed the technical foundations, proven the feasibility and motivated further investigations on dynamic, flexible teaming under partial workspace observability.

7.2. Future Work

To conclude with, Figure 7.1 summarises and adds to the suggested directions of future work mentioned in Section 7.1. A major limitation of the approach is that human actions that are not in line with the task model cause system failure – the system can neither detect, nor recover from situations in which priorly achieved task progress was undone, in which parts were placed at a wrong location etc. An initial step to relax this cooperative worker assumption would be to allow the team mental model state machines to exit the DONE state if the system detects that operation postconditions are no longer satisfied. Further error handling capabilities could be integrated by (i) adding an architectural component to provide state machine activities for managing different error cases (ii) and extending the coordination algorithm update procedure towards also reasoning about errors rather than merely about task progress (Section 5.3.3). Furthermore, the algorithm for task advancement (Section 5.3.2) could be revised to consider parameter grounding: Currently, the approach assumes unambiguously and fully parametrised operations – in particular, several parts of the same sort are implicitly identified by their precise initial locations and goal positions during the task modelling step. Making task allocation decisions by choosing a favourable combination of an operation and parts of the required type would relax these requirements: A parameter grounding step to select a concrete

part instance that an operation is applied to would render task modelling more flexible by accepting approximate part source locations as e.g. on a conveyer belt or a parts bin.

Aside from these conceptual extensions, limitations pointed out by the experiments can be addressed as follows: A pragmatic starting point for more comprehensive considerations regarding human cognitive workload is to adjust the Communicator implementation to constrain the number of messages that are presented to the worker at a time. Recent studies on communication planning as e.g. conducted by Unhelkar et al. [123] may provide further guidance towards balanced, two-way communication. An augmented reality (AR) approach could moreover be used as an intuitive communication channel. AR can be used to replace the visual cues of messages (Figure 6.3) by projecting the information directly into workers' fields of view on the one hand – on the other hand, projections might also be used to relax the assumption of workers knowing the task model precisely by investigating worker assistance with adaptive presentation of assembly instructions [7]. The issue of human idle times due to prolonged robot perception at the end of a task can be addressed by implementing means to extract more information from the given, limited sensor data: To this end, it may be beneficial to leave the top view onto the scene and enable enhanced active vision to increase the robot field of view. Furthermore, the development of a novel motion planning approach with the aim of increasing the information gain during each motion of skill execution is proposed – in this context, object certainty values (Section 4.2.3) can directly be used as an optimisation criterion to decide which parts to look at to maximally reduce uncertainty of the world model.

Especially the objective, quantitative assessment of the approach was based on notable simplifying assumptions to render simulation of dynamic human behaviour tractable. The human was particularly assumed to only use one hand to establish a baseline of equal partnership with a single manipulator. Extending the system algorithms towards using multiple robot arms is therefore an important direction of future investigations – showing scalability of the approach when teaming up a multi-arm robot with more complex models of dynamic human behaviour is a crucial next step to prepare studies that implement the domains of concrete shop floor scenarios. Ultimately, applicability to applications outside the laboratory does, of course, require a safety concept. Strictly speaking, each task model represents an application that must be certified anew in line with the regulations of ISO/TS 15066 [56]. First steps towards modifying applications within bounds that do not entail the need for a re-certification haven been made by Brandstötter et al. [21]. Investigations on the integration of similar concepts with the task modelling approach proposed in this thesis are a promising direction for future research. Furthermore, methods for safe interaction with points of contact to the previously proposed AR worker support system [125] as well as with LIDAR-based human tracking [106] as used in the prototype offer promising ideas for safety in flexible human-robot teams.

In addition to the above concrete suggestions for enhancements of the approach, the following more general future research directions are suggested: Currently, a skill according to the framework in Section 3.1 can only encode a fixed robot role in collaborative operations (e.g. holding a base part while a human partner assembles further components to it) – it would be interesting to investigate a concept of *skills with roles* where the skill

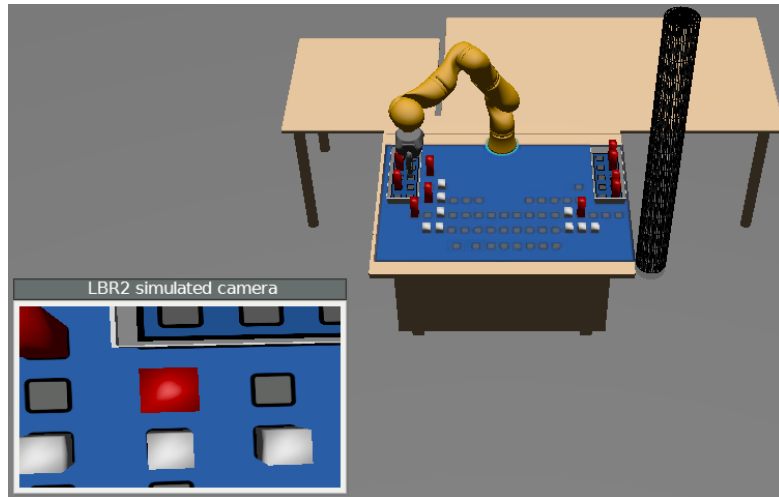
definition comprises all feasible roles. Role assignment to agents would then take place at execution time as a part of coordination. The proposed concept allows for strongly altering robot behaviour by modifying the task allocation metrics and exchanging the mental model state machines. Several teaming modes as e.g. human commanding or negotiation of task allocation for each operation can be realised this way. Therefore, this work not only enables future user studies on flexible teaming, but also a comparative evaluation of different teaming schemes. Finally, a general framework for simulating realistic, dynamic decisions of humans in tasks could generally contribute to the future development of dynamic teaming approaches. In this work, a set of human preferences was defined based on plausibility considerations – these preferences and particularly the likelihood of respective strategies to match real human workers’ habits should be further substantiated. To this end, a set of human joint action observations in predefined benchmark tasks could be clustered into human strategies similar to the approach proposed by Nikolaidis et al. [88]. A mapping to more general, abstract interaction patterns could then provide a benchmark database of tasks with associated human strategies to be pursued when simulating human-robot teaming.

Complementary Evaluation Data and Parametrisation Details

Simulation System

The simulation system (Figure A.1) establishes a digital twin of the hardware setup to simulate dynamic human-robot teaming workflows. To this end, robot motion as well as sensor data are simulated. Robot vision is particularly enabled by synthetic camera images. Basic human motion in the workspace is emulated by animating movement of a pillar on the roadmap specified with the domain definition (Section 4.2.1). Roadmap edges are defined to run parallel to the boundaries of workbenches within a distance of 20 cm. This abstract representation of workers realises simulated human decisions generated according to Section 6.2.2 by implementing basic actions of a domain and carrying out skills in analogy to a Skill Execution Engine implementation for the robot.

Figure A.1.: A digital twin of the prototype hardware setup enables simulation of arbitrary dynamic human-robot teaming workflows. Aside from human and robot motion, the system generates synthetic camera images (bottom left).



Critical trust factor λ_{crit}

Let the incremental certainty calculation according to Equation 4.15 be issued in fixed intervals of $\Delta t = 0.33$ s. Furthermore, let \bar{D}_R^τ denote the average operation duration across all operations within a task when carried out by the robot. We can then estimate that approximately \bar{D}_R^τ seconds pass between two robot decisions. World model ageing ensures a certainty value $\mathcal{C}_{t+\bar{D}_R^\tau}(e) = 0$ after a timespan \bar{D}_R^τ for some part e under constant worst-case human influence likelihood ($\mathcal{HI} = 1$) for

$$\lambda_{\text{crit}} = \frac{\Delta t}{\bar{D}_R^\tau}. \quad (\text{A.1})$$

Average operation durations \bar{D}_R^τ that were measured during five simulation runs per task are reported in Table A.1. Strictly speaking, an individual λ_{crit} value should be calculated for each task. However, as all \bar{D}_R^τ values are similar and lie around 10 s, a unified λ_{crit} value for all tasks was obtained by averaging the values in Table A.1 and substituting the result for \bar{D}_R^τ in Equation A.1.

Human picking and placing durations t_{pick} and t_{place}

The durations for picking and placing parts in the human simulation were roughly estimated using MTM-1 motion elements to approximate realistic human working pace. The sequences used were

$$\text{Rd}^* \text{B} - \text{G1A} - \text{Md}^* \text{B1.1}$$

for picking and

$$\text{Md}^* \text{B1.1} - \text{P1SE} - \text{RL1} - \text{Rd}^* \text{E}$$

for placing parts with a mass below 1.1 kg. The reaching and moving distance d^* was not calculated precisely for each simulated operation but approximated by the mean distance d^* of part start and goal positions from the human roadmap. This simplification is viable as we are interested in the overall teaming performance within tasks which is not affected by precise pick/place durations differing by fractions of a second. To determine d^* , the mean part distance \bar{d}_H^{reach} per task was measured based on operation pre- and postconditions for each task (Table A.1). Then, d^* was set to the mean value of these task-dependent values.

	Task A	Task B	Task C	Task D	Task E Var. 1	Task E Var. 2
$v_{\text{max}}^R [\frac{\text{m}}{\text{s}}]$	0.62	0.70	0.73	0.73	0.68	0.69
$\bar{D}_R^\tau [\text{s}]$	9.1 ± 1.5	10.4 ± 1.4	10.0 ± 1.4	9.5 ± 1.5	10.4 ± 1.9	10.7 ± 2.2
$\bar{d}_H^{\text{reach}} [\text{cm}]$	51.6 ± 5.7	47.1 ± 10.8	32.8 ± 7.6	32.8 ± 7.5	47.4 ± 9.4	43.2 ± 11.4

Table A.1.: Complementary Evaluation Data: Maximum robot TCP velocity v_{max}^R , mean robot operation duration \bar{D}_R^τ and mean human part reach distance \bar{d}_H^{reach} with standard deviation per task

Reference data acquisition

Reference values for each task when executed solely by a human agent (D_H) or by the robot (D_R) are summarised in Table A.2. A total of five simulation runs were carried out for each task and human strategy that involves partly randomised decisions (spatial strategy ‘random’). Respective reference values are therefore reported including their standard deviations across these iterations. A single simulation run is sufficient for all other strategies as they are deterministic.

World model ageing is not applied at all as long as there is no human presence in the workspace (cf. Equation 4.15, where \mathcal{HI} stays 0 in this case). This means that all objects are kept in the world model with a constant certainty value of 1. This leads to an equal weight of all operations when taking task allocation decisions in Algorithm 2. As outlined on page 85, the algorithm then decides randomly – this results in randomized robot action as long as there is no human present. Reference durations D_R were therefore also gathered by averaging across five simulation runs per task model.

	D_H [s]						D_R [s]	D_{opt} [s]
	‘none’	‘depth’	‘breadth’	‘none’	‘depth’	‘breadth’		
	‘random’	‘random’	‘random’	‘nearest’	‘nearest’	‘nearest’		
Task A	96.6 ± 0.2	96.4 ± 0.1	96.8 ± 0.1	96.5	96.5	96.3	181.5 ± 0.5	60.1
Task B	118.9 ± 0.2	118.8 ± 0.2	118.7 ± 0.2	118.0	118.6	118.2	208.7 ± 1.2	74.1
Task C	114.1 ± 1.2	112.2 ± 0.3	114.4 ± 1.7	111.0	111.0	110.9	199.8 ± 0.6	70.2
Task D	98.7 ± 2.1	95.5 ± 3.2	102.1 ± 4.4	87.8	87.9	88.3	190.3 ± 3.3	57.7
Task E Var. 1	160.8 ± 2.6	174.8 ± 0.1	157.9 ± 0.1	164.8	174.5	157.9	311.2 ± 3.5	n/a
Task E Var. 2	148.2 ± 2.6	144.4 ± 1.2	148.8 ± 1.8	142.3	142.2	142.2	322.0 ± 3.3	n/a

Table A.2.: Complementary Evaluation Data: Human-only task duration D_H per simulation strategy, robot only task duration D_R and optimal task duration D_{opt} resulting from capability-based optimisation

The optimal duration D_{opt} of each task when using both teammates to capacity was determined by calculating optimal schedules. For this purpose, the approach to capability-based human-robot task allocation by genetic optimisation as described by Beumelburg [16] was adapted to match the boundary conditions of this work in a student project [15]. In contrast to the original approach the modified implementation supports human- and robot-exclusive workspaces, collaborative operations and TYPE 2 operation decompositions for object handovers. Human and robot were assumed equally suited for all types of operations. Yet, shared tool access as necessary for Tasks E Var. 1 and Var. 2 was not implemented, i.e. there are no reference values available for these tasks. The optimisation tool implementation relies on the same human movement speed v^H and identical action durations t_{pick} , t_{place} and t_{shake} as used by the simulation system. These values were used to determine human operation durations in line with the simulation by calculating travelling distances on the roadmap, summing up durations for picking and placing etc. These analytically determined durations do, however, not match those of identical operations in the simulation precisely: Durations measured by the simulation system include a minor overhead originating from imprecisions that are caused by the

animation of human and robot motion. Compared to the optimisation tool, the simulation system may thus slightly overestimate operation durations. In consequence, the maximum possible speed up Σ_{opt} that compares human durations D_H from simulation runs with optimal durations D_{opt} from the optimisation tool may overestimate the cooperative potential of tasks within the bounds of this measurement error.

Experimental Protocol

The evaluation results in Section 6.2.5 summarize data from the following experiments:

- **Experiment 1:** Tasks A - D were tested with each combination of the six human strategies $\{\text{'depth-first'}, \text{'breadth-first'}, \text{'none'}\} \times \{\text{'nearest'}, \text{'random'}\}$ and the trust factor values in $\{\lambda_1, \dots, \lambda_5\}$ for $F_2^{\mathcal{H}\mathcal{I}}$ (with LIDAR data). Task execution was simulated ten times for each combination of task, human strategy and trust factor value. This leads to a total of $4 \cdot 6 \cdot 5 \cdot 10 = \mathbf{1200}$ **cooperative workflows** (data used for compiling Figure 6.5, Table 6.3, Figure 6.6 ‘with LIDAR’, Figure 6.7 ‘with LIDAR’ and Table 6.4 ‘without tool sharing’).
- **Experiment 2:** The second set of simulation runs was identical to Experiment 1, but used the interaction indicator set $F_1^{\mathcal{H}\mathcal{I}}$ (without LIDAR data) instead. This adds data from another **1200 simulation runs** to the experimental results (data used for compiling Figure 6.6 ‘without LIDAR’ and Figure 6.7 ‘without LIDAR’).
- **Experiment 3:** Another experiment targeted Task E, Variants 1 and 2 that involve sharing of a tool. This task was considered separately as reference data on optimal schedules was not available (see above). Teamwork based on both task models was simulated for all human strategies and trust factor values with $F_2^{\mathcal{H}\mathcal{I}}$ and ten iterations per combination, thus contributing $2 \cdot 6 \cdot 5 \cdot 10 = \mathbf{600}$ **simulated workflows** (data used for compiling Figure 6.8, Table 6.4 ‘with tool sharing’ and Table 6.5).
- **Experiment 4:** System performance with fully randomised robot decisions was measured for Tasks A - D and all human simulation strategies. The trust factor value was irrelevant for this experiment as certainty did not influence robot decisions at all. Again, 10 iterations per task model and human strategy were executed. The experiment thus resulted in $4 \cdot 6 \cdot 10 = \mathbf{240}$ **teaming processes** (data used for compiling Figure 6.6 and Figure 6.7 ‘random’).

List of Tables

1.1. Classification of the interaction targeted by this work	7
2.1. Properties of task models for human-robot teaming	18
2.2. Decision criteria for static/dynamic task allocation	21
2.3. Properties of dynamic task allocation approaches	24
3.1. Benchmark domain basic action input parameters	40
3.2. Benchmark domain skills	42
5.1. Conceptually necessary communication requests	82
6.1. Benchmark tasks start and goal states with reference data	100
6.2. Relevant parameter values for experimental evaluation	102
6.3. Error rates and human idle times depending on the trust factor	107
6.4. Human idle times due to waiting for resource availability	111
6.5. Robot error rates in Task E with shared access to a tool	111
A.1. Maximum robot TCP velocity, mean robot operation duration and human part reach distance per task	124
A.2. Human-only task duration and optimal task duration	125

List of Figures

1.1. Lightweight robot examples	2
1.2. Examples for modes of co-working in flexible teams	4
1.3. Human-robot coexistence, cooperation and collaboration	8
1.4. Flexible human-robot teaming	9
1.5. Limited Hardware Setup	12
2.1. Examples of task models	16
3.1. Task modelling framework overview	30
3.2. Skill graph structure	32
3.3. Example basic actions	35
3.4. Skill graph example	38
3.5. Benchmark domain	39
3.6. Workspace Layouts	44
3.7. Graphical Editor for Task Modelling	45
3.8. Skill parametrisation GUI elements	46
3.9. Task progress estimate update	48
4.1. Certainty for human-aware world modelling	52
4.2. World model initialization	53
4.3. World model update example	55
4.4. Human workspace roadmap	57
4.5. Variables and definitions for defining the human handling area	59
4.6. Point-based accessibility	60
4.7. Accessibility scalar fields	61
4.8. Qualitative certainty function profiles	63
5.1. Coordination process	68
5.2. Team mental model concept	69
5.3. Agent capability model based on human and robot range of action	72
5.4. Interaction category semantics	73
5.5. Flexible communication patterns	75

5.6. Preemptive team mental model state machines	78
5.7. System architecture	80
5.8. Knowledge update step example	87
6.1. Prototype system setup	94
6.2. Calibration coordinate frames and tooling	96
6.3. Smartphone-based prototype communicator implementation	98
6.4. Benchmark task precedence graphs	101
6.5. Relation of optimal and cooperative speedup, influence of ageing strategy	105
6.6. Influence of sensor data	108
6.7. Speedup per task model and human strategy	110
6.8. Influence of task model structure on system performance	110
7.1. Proposed directions of future work	119
A.1. Simulation system	123

Bibliography

- [1] Iina Aaltonen, Timo Salmi, and Ilari Marstio. Refining levels of collaboration to support the design and evaluation of human-robot interaction in the manufacturing industry. *Procedia CIRP*, 72:93–98, 2018.
- [2] Arash Ajoudani, Andrea Maria Zanchettin, Serena Ivaldi, Alin Albu-Schäffer, Kazuhiro Kosuge, and Oussama Khatib. Progress and prospects of the human-robot collaboration. *Autonomous Robots*, 42(5):957–975, 2018.
- [3] R. Alami, M. Warnier, J. Guitton, S. Lemaignan, and E. A. Sisbot. When the robot considers the human... In *Proceedings of the 15th International Symposium on Robotics Research*, 2011.
- [4] Rasmus Hasle Andersen, Lars Dalgaard, Anders Billeso Beck, and John Hallam. An architecture for efficient reuse in flexible production scenarios. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pages 151–157, Gothenburg, 2015.
- [5] Rasmus Hasle Andersen, Thomas Solund, and John Hallam. Definition and initial case-based evaluation of hardware-independent robot skills for industrial robotic co-workers. In *International Symposium on Robotics (ISR)*, pages 1–7, Munich, 2014.
- [6] E. A. Ashcroft. Proving assertions about parallel programs. *Journal of Computer and System Sciences*, 10(1):110–135, 1975.
- [7] A Bannat, Frank Wallhoff, G Rigoll, F Friesdorf, H Bubb, Sonja Stork, H Müller, A Schubö, M Wiesbeck, and M Zäh. Towards Optimal Worker Assistance: A Framework for Adaptive Selection and Presentation of Assembly Instructions. In *Proceedings of the 1st International Workshop on Cognition for Technical Systems (Cotesys)*, 2008.
- [8] Timo Bänziger, Andreas Kunz, and Konrad Wegener. Optimizing human-robot task allocation using a simulation tool based on standardized work descriptions. *Journal of Intelligent Manufacturing*, 2018.

- [9] Jimmy Baraglia, Maya Cakmak, Yukie Nagai, Rajesh PN Rao, and Minoru Asada. Efficient human-robot collaboration: When should a robot take initiative? *The International Journal of Robotics Research*, 36(5-7):563–579, 2017.
- [10] Andrea Bauer, Dirk Wollherr, and Martin Buss. Human-robot collaboration: a survey. *International Journal of Humanoid Robotics*, 5(01):47–66, 2008.
- [11] Jenay M. Beer, Arthur D. Fisk, and Wendy A. Rogers. Toward a Framework for Levels of Robot Autonomy in Human-Robot Interaction. *Journal of Human-Robot Interaction*, 3(2):74, 2014.
- [12] Roland Behrens, José Saenz, Christian Vogel, and Norbert Elkmann. Upcoming technologies and fundamentals for safeguarding all forms of human-robot collaboration. In *8th International Conference Safety of Industrial Automated Systems (SIAS)*, pages 18–23, Königswinter, 2015.
- [13] Manfred Bender, Martin Braun, Peter Rally, and Oliver Scholtz. Lightweight robots in manual assembly – best to start simply! Technical report, Fraunhofer Institute for Industrial Engineering IAO, Stuttgart, 2016.
- [14] Julia Berg and Gunther Reinhart. An Integrated Planning and Programming System for Human-Robot-Cooperation. *Procedia CIRP*, 63:95–100, 2017.
- [15] Daniel Bergmann. *Statische Aufgabenteilung für die Mensch-Roboter-Kooperation (B.Sc. thesis)*. University of Bayreuth, 2019.
- [16] Katharina Beumelburg. *Fähigkeitsorientierte Montageablaufplanung in der direkten Mensch-Roboter-Kooperation*. PhD thesis, Universität Stuttgart, 2005.
- [17] Rainer Bischoff, Arif Kazi, and Markus Seyfarth. The MORPHA style guide for icon-based programming. In *IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, pages 482–487, Berlin, 2002.
- [18] Sebastian Blankemeyer, Tobias Recker, Tobias Stuke, Jens Brokmann, Markus Geese, Michael Reiniger, Dennis Pischke, Assem Oubari, and Annika Raatz. A method to distinguish potential workplaces for human-robot collaboration. *Procedia CIRP*, 76:171–176, 2018.
- [19] R. Bokranz and K. Landau. *Handbuch Industrial Engineering: Produktivitätsmanagement mit MTM. Band 1: Konzept*. Schäffer-Poeschel, Stuttgart, 2 edition, 2012.
- [20] Jeffrey M. Bradshaw, Paul J. Feltovich, Matthew J. Johnson, Larry Bunch, Maggie R. Breedy, Tom Eskridge, Hyuckchul Jung, James Lott, and Andrzej Uszok. Coordination in Human-Agent-Robot Teamwork. In *International Symposium on Collaborative Technologies and Systems*, pages 467–476, Irvine, 2008.

-
- [21] Mathias Brandstötter, Titanilla Komenda, Fabian Ranz, Philipp Wedenig, Hubert Gattringer, Lukas Kaiser, Guido Breitenhuber, Andreas Schlotzhauer, Andreas Müller, and Michael Hofbaur. Versatile Collaborative Robot Applications Through Safety-Rated Modification Limits. In *International Conference on Robotics in Alpe-Adria Danube Region (RAAD)*, pages 438–446, Kaiserslautern, 2019.
 - [22] John Brown. Some Tests of the Decay Theory of Immediate Memory. *Quarterly Journal of Experimental Psychology*, 10(1):12–21, feb 1958.
 - [23] Nakul Gopalan Brown and Stefanie Tellex. Modeling and Solving Human-Robot Collaborative Tasks Using POMDPs. In *RSS Workshop on Model Learning for Human-Robot Communication*, 2015.
 - [24] Baptiste Busch, Marc Toussaint, and Manuel Lopes. Planning Ergonomic Sequences of Actions in Human-Robot Interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1916–1923, Brisbane, 2018.
 - [25] Andrea Casalino, Andrea Maria Zanchettin, Luigi Piroddi, and Paolo Rocco. Optimal Scheduling of Human-Robot Collaborative Assembly Operations With Time Petri Nets. *IEEE Transactions on Automation Science and Engineering*, pages 1–15, 2019.
 - [26] Fei Chen, Kosuke Sekiyama, Ferdinando Cannella, and Toshio Fukuda. Optimal subtask allocation for human and robot collaboration within hybrid assembly system. *IEEE Transactions on Automation Science and Engineering*, 11(4):1065–1075, 2014.
 - [27] Min Chen, Stefanos Nikolaidis, Harold Soh, David Hsu, and Siddhartha Srinivasa. Planning with Trust for Human-Robot Collaboration. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 307–315, New York, 2018.
 - [28] E. G. Coffman, M. Elphick, and A. Shoshani. System Deadlocks. *ACM Computing Surveys (CSUR)*, 3(2):67–78, jun 1971.
 - [29] Konstantinos Daniilidis. Hand-Eye Calibration Using Dual Quaternions. *International Journal of Robotics Research*, 18(3):286–298, 1999.
 - [30] Kouros Darvish, Barbara Bruno, Enrico Simetti, Fulvio Mastrogiovanni, and Giuseppe Casalino. Interleaved Online Task Planning, Simulation, Task Allocation and Motion Control for Flexible Human-Robot Cooperation. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 58–65, Nanjing, 2018.
 - [31] Urban Daub, Sarah Gawlick, and Florian Blab. Ergonomic Workplace Design - Musculoskeletal Relief Principles Deriving from the Exercise, Sports and Human Factor Sciences. Technical report, Fraunhofer Institute for Manufacturing Engineering and Automation (IPA), Stuttgart, 2018.

- [32] Lavindra de Silva, Raphael Lallement, and Rachid Alami. The HATP hierarchical planner: Formalisation and an initial study of its usability and practicality. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6465–6472, Hamburg, sep 2015.
- [33] Sandra Devin and Rachid Alami. An implemented theory of mind to improve human-robot shared plans execution. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 319–326, Christchurch, 2016.
- [34] EN ISO 10218-1:2011. Robots and robotic devices – Safety requirements for industrial robots – Part 1: Robots, 2011.
- [35] EN ISO 10218-2:2011. Robots and robotic devices – Safety requirements for industrial robots – Part 2: Robot systems and integration, 2011.
- [36] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, Portland, 1996.
- [37] Pedro Ferreira, Stefanos Doltsinis, and Niels Lohse. Symbiotic Assembly Systems – A New Paradigm. *Procedia CIRP*, 17:26–31, 2014.
- [38] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3-4):143–166, 2003.
- [39] Mary Ellen Foster, Manuel Giuliani, Thomas Müller, Markus Rickert, Alois Knoll, Wolfram Erlhagen, Estela Bicho, Nzoji Hipólito, and Luis Louro. Combining Goal Inference and Natural-Language Dialogue for Human-Robot Joint Action. In *Proceedings of the International Workshop on Combinations of Intelligent Methods and Applications, European Conference on Artificial Intelligence*, Patras, 2008.
- [40] Matthew Gombolay, Anna Bair, Cindy Huang, and Julie Shah. Computational design of mixed-initiative human–robot teaming that considers human factors: situational awareness, workload, and workflow preferences. *The International Journal of Robotics Research*, 36(5-7):597–617, 2017.
- [41] Michael A. Goodrich and Alan C. Schultz. Human-Robot Interaction: A Survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.
- [42] Patrik Gustavsson, Magnus Holm, Anna Syberfeldt, and Lihui Wang. Human-robot collaboration – towards new metrics for selection of communication technologies. *Procedia CIRP*, 72:123–128, 2018.
- [43] Roni-Jussi Halme, Minna Lanz, Joni Kämäräinen, Roel Pieters, Jyrki Latokartano, and Antti Hietanen. Review of vision-based safety systems for human-robot collaboration. *Procedia CIRP*, 72:111–116, 2018.

-
- [44] Takuma Hamabe, Hiraki Goto, and Jun Miura. A programming by demonstration system for human-robot collaborative assembly tasks. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1195–1201, Zhuhai, 2015.
 - [45] David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.
 - [46] Tsutomu Hasegawa, Takashi Suehiro, and Kunikatsu Takase. A Model-Based Manipulation System with Skill-Based Execution. *IEEE Transactions on Robotics and Automation*, 8(5):535–544, 1992.
 - [47] Kelsey P. Hawkins, Shray Bansal, Nam N. Vo, and Aaron F. Bobick. Anticipating human actions for collaboration in the presence of task and sensor uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2215–2222, Hong Kong, 2014.
 - [48] Bradley Hayes and Brian Scassellati. Effective robot teammate behaviors for supporting sequential manipulation tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6374–6380, Hamburg, 2015.
 - [49] Bradley Hayes and Brian Scassellati. Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5469–5476, Stockholm, 2016.
 - [50] Sue Hignett and Lynn McAtamney. Rapid Entire Body Assessment (REBA). *Applied Ergonomics*, 31(2):201–205, 2000.
 - [51] Guy Hoffman. Evaluating Fluency in Human–Robot Collaboration. *IEEE Transactions on Human-Machine Systems*, 49(3):209–218, 2019.
 - [52] Guy Hoffman and Cynthia Breazeal. Collaboration in Human-Robot Teams. In *AIAA 1st Intelligent Systems Technical Conference*, Reston, 2004.
 - [53] Luiz S. Homem de Mello and Arthur C. Sanderson. And/OR graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199, 1990.
 - [54] Luiz S. Homem de Mello and Arthur C. Sanderson. Representations of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation*, 7(2):211–227, 1991.
 - [55] ISO/TR 7250-2:2010/AMD1:2013. Basic human body measurements for technological design – Part 2: Statistical summaries of body measurements from national populations, 2010.
 - [56] ISO/TS 15066:2016. Robots and robotic devices - Collaborative robots, 2016.

- [57] P. Jiménez. Survey on assembly sequencing: a combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing*, 24(2):235–250, 2013.
- [58] Lars Johannsmeier and Sami Haddadin. A Hierarchical Human-Robot Interaction-Planning Framework for Task Allocation in Collaborative Industrial Assembly Processes. *IEEE Robotics and Automation Letters*, 2(1):41–48, 2017.
- [59] Catholijn M Jonker, M Birna van Riemsdijk, and Bas Vermeulen. Shared Mental Models: A Conceptual Analysis. In *Proceedings of the 6th International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*, pages 132–151, Lyon, 2010.
- [60] Christian Juelg, Andreas Hermann, Arne Roennau, and Rüdiger Dillmann. Efficient, collaborative screw assembly in a shared workspace. In *Advances in Intelligent Systems and Computing*, volume 867, pages 837–848. Springer Verlag, 2019.
- [61] Krishnanand N. Kaipa, Carlos W. Morato, and Satyandra K. Gupta. Design of Hybrid Cells to Facilitate Safe and Efficient Human–Robot Collaboration During Assembly Operations. *Journal of Computing and Information Science in Engineering*, 18(3):031004, jun 2018.
- [62] Erez Karpas, Steven James Levine, Peng Yu, and Brian C Williams. Robust Execution of Plans for Human-Robot Teams. In *International Conference on International Conference on Automated Planning and Scheduling (ICAPS)*, pages 342–346, Jerusalem, 2015.
- [63] Johan Kildal, Alberto Tellaeche, Izaskun Fernández, and Iñaki Mautua. Potential users’ key concerns and expectations for the adoption of cobots. *Procedia CIRP*, 72:21–26, 2018.
- [64] H. Kimura, T. Horiuchi, and K. Ikeuchi. Task-model based human robot cooperation using vision. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 701–706, Kyongju, 1999.
- [65] Hema S. Koppula, Ashesh Jain, and Ashutosh Saxena. Anticipatory planning for human-robot teams. In *International Symposium on Experimental Robotics (ISER)*, pages 453–470. Springer, 2016.
- [66] Martin Kraft and Markus Rickert. How to teach your robot in 5 minutes: Applying UX paradigms to human-robot-interaction. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 942–949, Lisboa, 2017.
- [67] S. Lallée, U. Pattacini, S. Lemaignan, A. Lenz, C. Melhuish, L. Natale, S. Skachek, K. Hamann, J. Steinwender, E. A. Sisbot, G. Metta, J. Guitton, R. Alami,

- M. Warnier, T. Pipe, F. Warneken, and P. F. Dominey. Towards a Platform-Independent Cooperative Human Robot Interaction System: III An Architecture for Learning and Executing Actions and Shared Plans. *IEEE Transactions on Autonomous Mental Development*, 4(3):239–253, 2012.
- [68] R. Lallement, L. de Silva, and R. Alami. HATP: An HTN Planner for Robotics. In *2nd ICAPS Workshop on Planning and Robotics*, Portsmouth, 2014.
- [69] Przemyslaw A. Lasota, Terrence Fong, and Julie A. Shah. A Survey of Methods for Safe Human-Robot Interaction. *Foundations and Trends in Robotics*, 5(3):261–349, 2017.
- [70] Johan Sund Laursen, Ulrik Pagh Schultz, and Lars Peter Ellekilde. Automatic error recovery in robot assembly operations using reverse execution. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1785–1792, Hamburg, 2015.
- [71] C. Lenz, A. Sotzek, T. Roder, H. Radrich, A. Knoll, M. Huber, and S. Glasauer. Human workflow analysis using 3D occupancy grid hand tracking in a human-robot collaboration scenario. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3375–3380, San Francisco, 2011.
- [72] Claus Lenz and Alois Knoll. Mechanisms and capabilities for human robot collaboration. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 666–671, Edinburgh, 2014.
- [73] Steven J. Levine and Brian C. Williams. Watching and Acting Together: Concurrent Plan Recognition and Adaptation for Human-Robot Teams. *Journal of Artificial Intelligence Research*, 63:281–359, 2018.
- [74] Beng-Chong Lim and Katherine J. Klein. Team mental models and team performance: a field study of the effects of team mental model similarity and accuracy. *Journal of Organizational Behavior*, 27(4):403–418, 2006.
- [75] Matthias Linsinger, Martin Sudhoff, Kai Lemmerz, Paul Glogowski, and Bernd Kuhlenkötter. Task-based Potential Analysis for Human-Robot Collaboration within Assembly Systems. In *Tagungsband des 3. Kongresses Montage Handhabung Industrieroboter*, pages 1–12, Erlangen, 2018.
- [76] Liliana Lo Presti and Marco La Cascia. 3D skeleton-based human action classification: A survey. *Pattern Recognition*, 53:130–147, 2016.
- [77] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.
- [78] Ilias El Makrini, Kelly Merckaert, Joris De Winter, Dirk Lefeber, and Bram Vanderborght. Task allocation for improved ergonomics in Human-Robot Collaborative Assembly. *Interaction Studies*, 20(1):102–133, 2019.

- [79] Thomas W. Malone and Kevin Crowston. What is coordination theory and how can it help design cooperative work systems? In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work - CSCW '90*, pages 357–370, New York, 1990.
- [80] Matthew T. Mason. Compliance and Force Control for Computer Controlled Manipulators. *IEEE Transactions on Systems, Man and Cybernetics*, 11(6):418–432, 1981.
- [81] João Costa Mateus, Dieter Claeys, Veronique Limère, Johannes Cottyn, and El-Houssaine Aghezaf. A structured methodology for the design of a human-robot collaborative assembly workplace. *The International Journal of Advanced Manufacturing Technology*, 102(5-8):2663–2681, jun 2019.
- [82] John E. Mathieu, Tonia S. Heffner, Gerald F. Goodwin, Eduardo Salas, and Janis A. Cannon-Bowers. The influence of shared mental models on team process and performance. *Journal of Applied Psychology*, 85(2):273–283, 2000.
- [83] Nikolaos Mavridis. A review of verbal and non-verbal human-robot interactive communication. *Robotics and Autonomous Systems*, 63:22–35, 2015.
- [84] George Michalos, Jason Spiliotopoulos, Sotiris Makris, and George Chryssolouris. A method for planning human robot shared tasks. *CIRP Journal of Manufacturing Science and Technology*, 22:76–90, aug 2018.
- [85] Gregoire Milliez, Raphael Lallement, Michelangelo Fiore, and Rachid Alami. Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 43–50, Christchurch, 2016.
- [86] Michela Dalle Mura and Gino Dini. Designing assembly lines with humans and collaborative robots: A genetic approach. *CIRP Annals*, 68(1):1–4, 2019.
- [87] Anja Naumann and Jörn Hurtienne. Benchmarks for intuitive interaction with mobile devices. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 401–402, Lisboa, 2010.
- [88] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. Efficient Model Learning from Joint-Action Demonstrations for Human-Robot Collaborative Tasks. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 189–196, Portland, 2015.
- [89] Stefanos Nikolaidis and Julie Shah. Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 33–40, Tokyo, 2013.

-
- [90] Nikolaos Nikolakis, Konstantinos Sipsas, Panagiota Tsarouchi, and Sotiris Makris. On a shared human-robot task scheduling and online re-scheduling. *Procedia CIRP*, 78:237–242, 2018.
 - [91] Xinwen Niu, Han Ding, and Youlun Xiong. A hierarchical approach to generating precedence graphs for assembly planning. *International Journal of Machine Tools and Manufacture*, 43(14):1473–1486, 2003.
 - [92] José Novoa, Jorge Wuth, Juan Pablo Escudero, Josué Fredes, Rodrigo Mahu, and Néstor Becerra Yoma. DNN-HMM based Automatic Speech Recognition for HRI Scenarios. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 150–159, New York, 2018.
 - [93] International Federation of Robotics. Robots and the Workplace of the Future (Positioning Paper), 2018.
 - [94] E. Orendt, M. Fichtner, and D. Henrich. Robot programming by non-experts: Intuitiveness and robustness of One-Shot robot programming. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 192–199, New York, 2016.
 - [95] A. K. Pandey and R. Alami. Mightability maps: A perceptual level decisional framework for co-operative and competitive human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5842–5848, Taipei, 2010.
 - [96] Chris Paxton, Andrew Hundt, Felix Jonathan, Kelleher Guerin, and Gregory D. Hager. CoSTAR: Instructing collaborative robots with behavior trees and vision. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 564–571, Singapore, 2017.
 - [97] Margaret Pearce, Bilge Mutlu, Julie Shah, and Robert Radwin. Optimizing Makespan and Ergonomics in Integrating Collaborative Robots Into Manufacturing Processes. *IEEE Transactions on Automation Science and Engineering*, 15(4):1772–1784, 2018.
 - [98] Mikkel Rath Pedersen, Lazaros Nalpantidis, Rasmus Skovgaard Andersen, Casper Schou, Simon Bøgh, Volker Krüger, and Ole Madsen. Robot skills for manufacturing: From concept to industrial deployment. *Robotics and Computer-Integrated Manufacturing*, 37:282–291, 2016.
 - [99] Alexander Perzylo, Markus Rickert, Bjoern Kahl, Nikhil Somani, Christian Lehmann, Alexander Kuss, Stefan Profanter, Anders Billesø Beck, Mathias Haage, Mikkel Rath Hansen, Malene Tofveson Nibe, Maximo A. Roa, Olof Sornmo, Sven Gestegard Robertz, Ulrike Thomas, Germano Veiga, Elin Anna Topp, Ingmar Kessler, and Marinus Danzer. SMERobotics: Smart Robots for Flexible Manufacturing. *IEEE Robotics & Automation Magazine*, 26(1):78–90, mar 2019.

- [100] Fabian Ranz, Vera Hummel, and Wilfried Sihn. Capability-based Task Allocation in Human-robot Collaboration. *Procedia Manufacturing*, 9:182–189, 2017.
- [101] Timothy J. Ricker, Evie Vergauwe, and Nelson Cowan. Decay theory of immediate memory: From Brown (1958) to today (2014). *Quarterly Journal of Experimental Psychology*, 69(10):1969–1995, oct 2016.
- [102] Michael Riedel, Eric M. Orendt, and Dominik Henrich. Sensor-Based Loops and Branches for Playback-Programmed Robot Systems. In *International Conference on Robotics in Alpe-Adria Danube Region (RAAD)*, pages 183–190, Torino, 2017.
- [103] Alessandro Roncone, Olivier Mangin, and Brian Scassellati. Transparent role assignment and task allocation in human robot collaboration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1014–1021, Singapore, 2017.
- [104] Gregory F. Rossano, Carlos Martinez, Mikael Hedelind, Steve Murphy, and Thomas A. Fuhlbrigge. Easy robot programming concepts: An industrial perspective. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1119–1126, Madison, 2013.
- [105] Leonel Roza, Sylvain Calinon, Darwin G. Caldwell, Pablo Jimenez, and Carme Torras. Learning Physical Collaborative Robot Behaviors From Human Demonstrations. *IEEE Transactions on Robotics*, 32(3):513–527, jun 2016.
- [106] Mohammad Safeea and Pedro Neto. Minimum distance calculation using laser scanner and IMUs for safe human-robot interaction. *Robotics and Computer-Integrated Manufacturing*, 58:33–42, aug 2019.
- [107] Nikolaos Sarafianos, Bogdan Boteanu, Bogdan Ionescu, and Ioannis A. Kakadiaris. 3D Human pose estimation: A review of the literature and analysis of covariates. *Computer Vision and Image Understanding*, 152:1–20, 2016.
- [108] Philip J. Schneider and David H. Eberly. *Geometric Tools for Computer Graphics*. Morgan Kaufmann, San Francisco, 2003.
- [109] D. Schröter, P. Jaschewski, B. Kuhrke, and A. Verl. Methodology to Identify Applications for Collaborative Robots in Powertrain Assembly. *Procedia CIRP*, 55:12–17, 2016.
- [110] Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. People Tracking with Mobile Robots Using Sample-Based Joint Probabilistic Data Association Filters. *The International Journal of Robotics Research*, 22(2):99–116, feb 2003.
- [111] Eugenio Sebastiani, Raphaël Lallement, Rachid Alami, and Luca Iocchi. Dealing with On-line Human-Robot Negotiations in Hierarchical Agent-Based Task Planner. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (CAPS)*, pages 549–557, Pittsburgh, 2017.

-
- [112] Julie Shah, James Wiken, Brian Williams, and Cynthia Breazeal. Improved human-robot team performance using chaski, a human-inspired plan execution system. In *International Conference on Human-Robot Interaction (HRI)*, pages 29–36, New York, 2011.
 - [113] Mili Shah, Roger D. Eastman, and Tsai Hong. An overview of robot-sensor calibration methods for evaluation of perception systems. In *Performance Metrics for Intelligent Systems (PerMIS) Workshop*, pages 15–20, New York, 2012.
 - [114] Aaron St. Clair and Maja Mataric. How Robot Verbal Feedback Can Improve Team Performance in Human-Robot Task Collaborations. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 213–220, Portland, 2015.
 - [115] Jochen J. Steil and Günter W. Maier. Kollaborative Roboter: universale Werkzeuge in der digitalisierten und vernetzten Arbeitswelt. In Günter W Maier, Gregor Engels, and Eckhard Steffen, editors, *Handbuch Gestaltung digitaler und vernetzter Arbeitswelten*, pages 1–24. Springer Berlin Heidelberg, 2018.
 - [116] Aaron Steinfeld, Odest Chadwicke Jenkins, and Brian Scassellati. The oz of wizard. In *ACM/IEEE International Conference on Human Robot Interaction (HRI)*, pages 101–108, La Jolla, USA, 2009.
 - [117] Franz Steinmetz, Annika Wollschlager, and Roman Weitschat. RAZER - A HRI for Visual Task-Level Programming and Intuitive Skill Parameterization. *IEEE Robotics and Automation Letters*, 3(3):1362–1369, 2018.
 - [118] U. Thomas, M. Barrenscheen, and F.M. Wahl. Efficient assembly sequence planning using stereographical projections of C-space obstacles. In *IEEE International Symposium on Assembly and Task Planning*, pages 96–102, Besancon, 2003.
 - [119] U. Thomas, B. Finkemeyer, T. Kroger, and F.M. Wahl. Error-tolerant execution of complex robot tasks based on skill primitives. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 3069–3075, Taipei, 2003.
 - [120] Ulrike Thomas, Gerd Hirzinger, Bernhard Rumpe, Christoph Schulze, and Andreas Wortmann. A new skill based robot programming language using UML/P Statecharts. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 461–466, Karlsruhe, 2013.
 - [121] Elin A. Topp and Henrik I. Christensen. Tracking for following and passing persons. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 70–76, Edmonton, Canada, 2005.
 - [122] Kirsten Tracht, Lars Funke, and Michael Schottmayer. Online-control of assembly processes in paced production lines. *CIRP Annals*, 64(1):395–398, 2015.

- [123] Vaibhav V. Unhelkar, Shen Li, and Julie A. Shah. Decision-making for bidirectional communication in sequential human-robot collaborative tasks. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 329–341, New York, 2020.
- [124] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248–266, 2018.
- [125] Christian Vogel, Christoph Walter, and Norbert Elkmann. Safeguarding and Supporting Future Human-robot Cooperative Manufacturing Processes by a Projection- and Camera-based Technology. *Procedia Manufacturing*, 11:39–46, jan 2017.
- [126] R. S. Wilcox, S. Nikolaidis, and J. A. Shah. Optimization of Temporal Dynamics for Adaptive Human-Robot Interaction in Assembly Manufacturing. In *Robotics: Science and Systems*, 2012.
- [127] H.A. Yanco and J. Drury. Classifying human-robot interaction: an updated taxonomy. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 2841–2846, The Hague, 2004.
- [128] Guang-Zhong Yang, Jim Bellingham, Pierre E. Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, Bradley J. Nelson, Brian Scassellati, Mariarosaria Taddeo, Russell Taylor, Manuela Veloso, Zhong Lin Wang, and Robert Wood. The grand challenges of Science Robotics. *Science Robotics*, 3(14), jan 2018.
- [129] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3D Object Recognition in Cluttered Scenes with Local Surface Features: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–87, nov 2014.
- [130] Sofya Zeylikman, Sarah Widder, Alessandro Roncone, Olivier Mangin, and Brian Scassellati. The HRC Model Set for Human-Robot Collaboration Research. *arXiv:1710.11211v2 [cs.RO]*, oct 2018.
- [131] Q. Zhang and R. Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2301–2306, Sendai, 2004.
- [132] Wei Zheng, Bo Wu, and Hai Lin. POMDP Model Learning for Human Robot Collaboration. *arXiv:1803.11300v1 [cs.HC]*, 2018.

Prior publications of the author (peer-reviewed)

- [133] Dominik Riedelbauch and Dominik Henrich. Fast Graphical Task Modelling for Flexible Human-Robot Teaming. In *50th International Symposium on Robotics (ISR)*, pages 420–425, Munich, 2018.
- [134] Dominik Riedelbauch and Dominik Henrich. Exploiting a Human-Aware World Model for Dynamic Task Allocation in Flexible Human-Robot Teams. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6511–6517, Montréal, 2019.
- [135] Dominik Riedelbauch, Stephan Schweizer, and Dominik Henrich. Skill Interaction Categories for Communication in Flexible Human-Robot Teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3810–3816, Macau, 2019.
- [136] Dominik Riedelbauch, Tobias Werner, and Dominik Henrich. Supporting a Human-Aware World Model Through Sensor Fusion. In *International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, pages 665–672, Torino, 2017.

Dominik Riedelbauch and Dominik Henrich. Coordinating Flexible Human-Robot Teams by Local World State Observation. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1000–1005, Lisbon, 2017.

Dominik Riedelbauch, Tobias Werner, and Dominik Henrich. Enabling Domain Experts to Model and Execute Tasks in Flexible Human-Robot Teams. In *Tagungsband des 2. Kongresses Montage Handhabung Industrieroboter*, pages 13–22, Bremen, 2017.

Tobias Werner, Dominik Riedelbauch, and Dominik Henrich. Design and Evaluation of a Multi-Agent Software Architecture for Risk-Minimized Path Planning in Human-Robot Workcells. In *Tagungsband des 2. Kongresses Montage Handhabung Industrieroboter*, pages 103–112, Bremen, 2017.

Prior publications of the author (other)

- [140] Dominik Riedelbauch, Johannes Hartwig, and Dominik Henrich. Enabling End-Users to Deploy Flexible Human-Robot Teams to Factories of the Future. In *IROS 2019 Workshop "Factory of the Future"*, Macau, 2019.

Dominik Riedelbauch and Dominik Henrich. Eine Frage der Abstimmung. In *handling* 7-8, pages 42–43. WEKA Business Medien, 2017.

Dominik Riedelbauch and Dominik Henrich. Koordinierung hybrider Mensch-Roboter-Teams. In Rainer Müller, Jörg Franke, Dominik Henrich, Bernd Kuhlentötter, Annika Raatz, and Alexander Verl, editors, *Handbuch Mensch-Roboter-Kollaboration*, chapter 5.6, pages 260–269. Carl Hanser Verlag, 2019.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet habe. Weiterhin erkläre ich, dass ich die Hilfe von gewerblichen Promotionsberatern bzw. –vermittlern oder ähnlichen Dienstleistern weder bisher in Anspruch genommen habe, noch künftig in Anspruch nehmen werde. Zusätzlich erkläre ich hiermit, dass ich keinerlei frühere Promotionsversuche unternommen habe.

Bayreuth, den 12. November 2020

Dominik Riedelbauch